Modicon M340 with Unity Pro

Counting Module BMX EHC 0200
User Manual

07/2012

35013355.06

www.schneider-electric.com

All pertinent state, regional, and local safety regulations must be observed when installing and using this product. For reasons of safety and to help ensure compliance with documented system data, only the manufacturer should perform repairs to components.

When devices are used for applications with technical safety requirements, the relevant instructions must be followed.

Failure to use Schneider Electric software or approved software with our hardware products may result in injury, harm, or improper operating results.

Failure to observe this information can result in injury or equipment damage.

Table of Contents

Safety Information

Important Information

NOTICE

Read these instructions carefully, and look at the equipment to become familiar with the device before trying to install, operate, or maintain it. The following special messages may appear throughout this documentation or on the equipment to warn of potential hazards or to call attention to information that clarifies or simplifies a procedure.

The addition of this symbol to a Danger safety label indicates that an electrical hazard exists, which will result in personal injury if the instructions are not followed.

This is the safety alert symbol. It is used to alert you to potential personal injury hazards. Obey all safety messages that follow this symbol to avoid possible injury or death.

⚠ DANGER
DANGER indicates an imminently hazardous situation which, if not avoided, **will result in** death or serious injury.

⚠ WARNING
WARNING indicates a potentially hazardous situation which, if not avoided, **can result in** death or serious injury.

⚠ CAUTION

CAUTION indicates a potentially hazardous situation which, if not avoided, **can result in** minor or moderate injury.

NOTICE

NOTICE is used to address practices not related to physical injury.

PLEASE NOTE

Electrical equipment should be installed, operated, serviced, and maintained only by qualified personnel. No responsibility is assumed by Schneider Electric for any consequences arising out of the use of this material.

A qualified person is one who has skills and knowledge related to the construction and operation of electrical equipment and its installation, and has received safety training to recognize and avoid the hazards involved.

About the Book

At a Glance

Document Scope

This manual describes the hardware and software implementation of counting module BMX EHC 0200 for Modicon M340 PLCs and X80 drops.

Validity Note

This documentation is valid from Unity Pro V7.0.

Product Related Information

⚠ WARNING

UNINTENDED EQUIPMENT OPERATION

The application of this product requires expertise in the design and programming of control systems. Only persons with such expertise should be allowed to program, install, alter, and apply this product.

Follow all local and national safety codes and standards.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

User Comments

We welcome your comments about this document. You can reach us by e-mail at techcomm@schneider-electric.com.

Introduction to the Counting Function

I

Subject of this Part

This part provides a general introduction to the counting function and the operating principles of the BMX EHC 0200.

What Is in This Part?

This part contains the following chapters:

General Information on the Counting Function

1

General Information on Counting Functions

At a Glance

The counting function enables fast counting using couplers, Unity Pro screens and specialized language objects. The general operation of expert modules also known as couplers is described in the section Presentation of the Counting Module Operation BMX EHC 0200.

In order to implement the counting, it is necessary to define the physical context in which it is to be executed (rack, supply, processor, modules etc.) and to ensure the software implementation *(see page 101)*.

This second aspect is performed from the different Unity Pro editors:

- in offline mode
- in online mode

Presentation of Counting Module

2

Subject of this Chapter

This chapter deals with the counting module BMX EHC 0200 of the Modicon M340 range.

What Is in This Chapter?

This chapter contains the following topics:

General Information about Counting Module

Introduction

Counting module is standard format module that enable pulses from a sensor to be counted at a maximum frequency of 60 KHz (BMX EHC 0200).

The BMX EHC 0200 module has 2 channels.

This module may be installed in any available slot in a Modicon M340 PLC station rack.

Sensors Used

The sensors used on each channel may be:

- 24 VDC two-wire proximity sensors
- Incremental signal encoders with 10/30 VDC output and push-pull outputs.

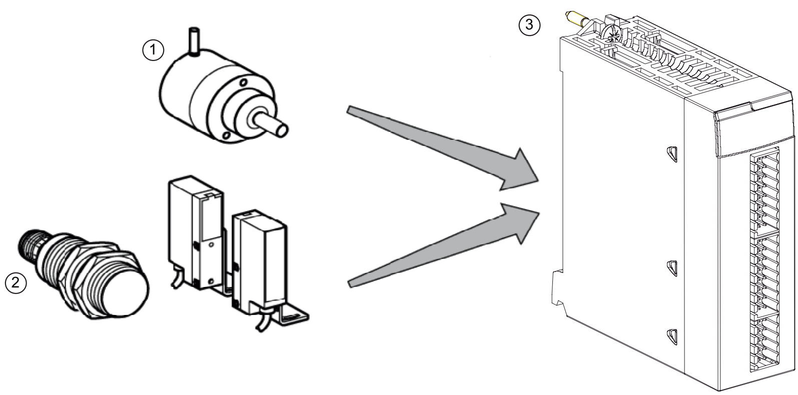
Illustration

The illustration below shows the following:

1) Incremental encoder

2) Proximity sensors

3) Counting module BMX EHC 0200



General Information about the Counting Module Operation

Introduction

The BMX EHC 0200 module is counting modules from the Modicon M340 modular PLC range. They support all Unity Pro software functionalities.

This module have:

- Counting-related functions (comparison, capture, homing, reset to 0)
- Event generation functions designed for the application program
- Outputs for actuator use (contacts, alarms, relays)

Characteristics

The main characteristics of BMX EHC 0200 module are as follows.

Application	Number of channels per module	Number of physical inputs per channel	Number of physical outputs per channel	Maximum frequency
• Counting • Downcounting • Up/Down counting • Measurement • Frequency meter • Frequency generator • Axis monitoring	2	6	2	60 KHz

Presentation of the BMX EHC 0200 Counting Module

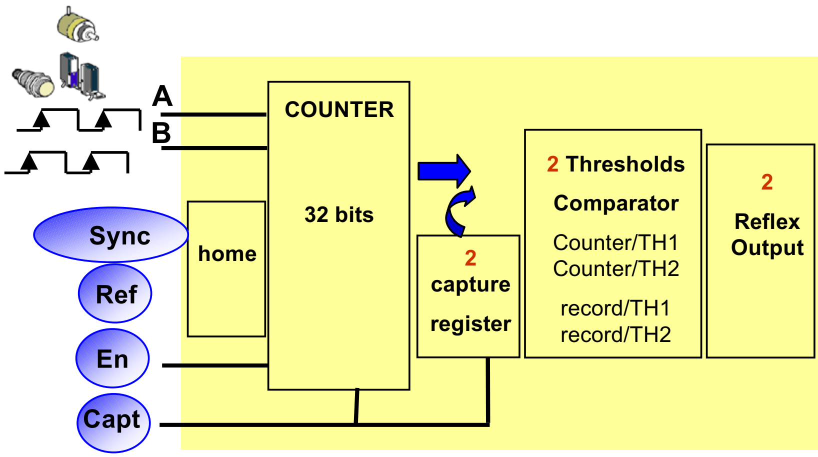
At a Glance

The BMX EHC 0200 counting module enables the counting or downcounting of pulses to be performed. It has the following functions:

- Enable
- Capture
- Comparison
- Homing or reset to 0
- 2 physical outputs

Structure of a counter channel

The following illustration shows the overall structure of a counter channel:



Modicon M340H (Hardened) Equipment

M340H

The Modicon M340H (hardened) equipment is a ruggedized version of M340 equipment. It can be used at extended temperatures (-25...70ºC) (-13...158ºF) and in harsh chemical environments.

This treatment increases the isolation capability of the circuit boards and their resistance to:
- condensation
- dusty atmospheres (conducting foreign particles)
- chemical corrosion, in particular during use in sulphurous atmospheres (oil, refinery, purification plant and so on) or atmospheres containing halogens (chlorine and so on)

The M340H equipment, when within the standard temperature range (0...60ºC) (32...140ºF), has the same performance characteristics as the standard M340 equipment.

At the temperature extremes (-25... 0ºC and 60... 70ºC) (-13...32ºF) and (140...158ºF) the hardened versions can have reduced power ratings that impact power calculations for Unity Pro applications.

If this equipment is operated outside the -25...70ºC (-13...158ºF) temperature range, the equipment can operate abnormally.

⚠ CAUTION

UNINTENDED EQUIPMENT OPERATION

Do not operate M340H equipment outside of its specified temperature range.

Failure to follow these instructions can result in injury or equipment damage.

Hardened equipment has a conformal coating applied to its electronic boards. This protection, when associated with appropriate installation and maintenance, allows it to be more robust when operating in harsh chemical environments.

Presentation of the Counting Module Operation

3

Overview of BMX EHC 0200 Module Functionalities

At a Glance

This part presents the different types of user applications for the BMX EHC 0200 module.

Measurement

The following table presents the measurement functionality for the BMX EHC 0200 module:

User application type	Mode
Speed measurement/stream measurement	Frequency
Random events monitoring	Event counting
Pulse evaluation/Speed control	Period measuring
Flow control	Ratio

Counting

The following table presents the counting functionality for the BMX EHC 0200 module:

User application type	Mode
Grouping	One shot counter
Level 1 packaging/labeling	Modulo loop counter
Level 2 packaging/labeling	Free large counter
Accumulator	Free large counter
Axis control	Free large counter

NOTE: In case of a user application such as level 1 packaging/labeling, the machine makes constant spacing between parts. In case of a user application such as level 2 packaging/labeling, the counting module learns the incoming edge of each part.

Frequency Generator

The following table presents the frequency generator functionality for the BMX EHC 0200 module:

User application type	Mode
Input frequency device	Pulse width modulation

Interface

The BMX EHC 0200 module may be interfaced with the following components:

- mechanical switch
- 24 VDC two-wire proximity sensor
- 24 VDC three-wire proximity sensor
- 10/30 VDC encoder with push-pull outputs

Counting Module BMX EHC 0200 Hardware Implementation

II

Subject of this Part

This part presents the hardware implementation of the BMX EHC 0200 counting module.

What Is in This Part?

This part contains the following chapters:

General Rules for Installing Counting Module BMX EHC 0200

4

Subject of this Chapter

This chapter presents the general rules for installing counting module BMX EHC 0200.

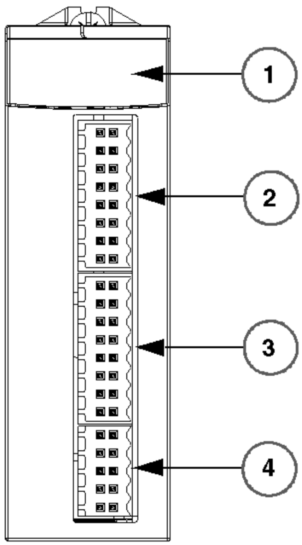
What Is in This Chapter?

This chapter contains the following topics:

Physical Description of the Counting Module

Illustration

The figure below present the counting module BMX EHC 0200 :



BMX EHC 0200

Physical Elements of the Modules

The table below presents the elements of the counting module BMX EHC 0200:

Number	Description
1	Module state LEDs: • State LEDs at module level • State LEDs at channel level
2	16-pin connector to connect the counter 0 sensors
3	16-pin connector to connect the counter 1 sensors
4	10-pin connector to connect: • Auxiliary outputs • Sensor power supplies

Accessories

The BMX EHC 0200 module requires the use of the following accessories:

- Two 16-pin terminal blocks
- One 10-pin terminal block
- One BMX XSP 0400/0600/0800/1200 electromagnetic compatibility kit *(see Modicon M340 Using Unity Pro, Processors, Racks, and Power Supply Modules, Setup Manual)*

NOTE: The two 16-pin connectors and the 10-pin connector are available under the reference BMX XTS HSC 20.

Fitting of Counting Modules

At a Glance

The counting modules are powered by the rack bus. The modules may be handled without turning off power supply to the rack, without damage or disturbance to the PLC.

Fitting operations (installation, assembly and disassembly) are described below.

Installation Precautions

The counting modules may be installed in any of the positions in the rack except for the first two (marked PS and 00) which are reserved for the rack's power supply module (BMX CPS ••••) and the processor (BMX P34 ••••) respectively. Power is supplied by the bus at the bottom of the rack (3.3 V and 24 V).

Before installing a module, you must take off the protective cap from the module connector located on the rack.

⚠ DANGER

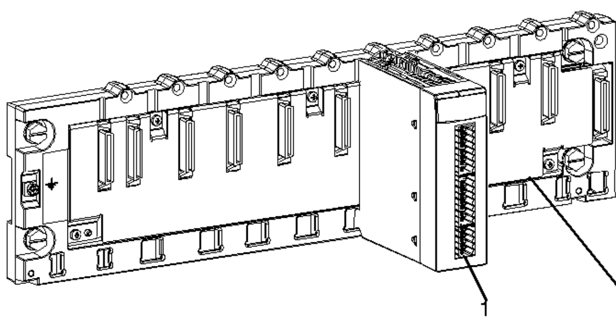
HAZARD OF ELECTRIC SHOCK

- disconnect voltage supplying sensors and pre-actuators before plugging / unplugging the terminal block on the module.
- remove the terminal block before plugging / unplugging the module on the rack.

Failure to follow these instructions will result in death or serious injury.

Installation

The diagram below shows counting module BMX EHC 0200 mounted on the rack:

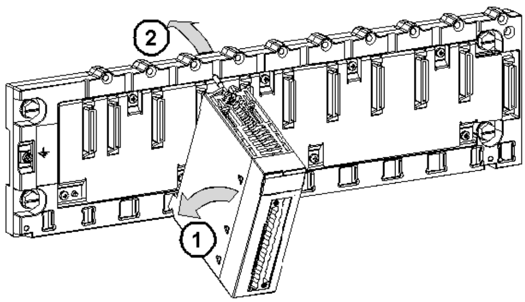
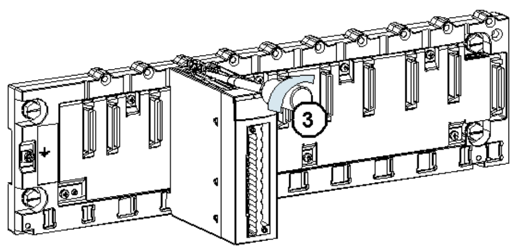


The following table describes the different elements which make up the assembly below:

Number	Description
1	BMX EHC 0200 counting module
2	Standard rack

Installing the Module on the Rack

The following table shows the procedure for mounting the counting module in the rack:

Step	Action	Illustration
1	Position the locating pins situated at the rear of the module (on the bottom part) in the corresponding slots in the rack. Note: Before positioning the pins, make sure you have removed the protective cover *(see Modicon M340 Using Unity Pro, Processors, Racks, and Power Supply Modules, Setup Manual)*.	Steps 1 and 2 
2	Swivel the module towards the top of the rack so that the module sits flush with the back of the rack. It is now set in position.	
3	Tighten the safety screw to ensure that the module is held in place on the rack. Tightening torque: Max. 1.5 N.m	Step 3 

Fitting 10-Pin and 16-Pin Terminal Blocks to a BMX EHC 0200 Counting Module

At a Glance

BMX EHC 0200 counting modules with 10-pin and 16-pin terminal block connections require either or both terminal blocks to be connected to the module. These fitting operations (assembly and disassembly) are described below.

Installing the 10-Pin and 16-Pin Terminal Blocks

⚠ DANGER

ELECTRICAL SHOCK

Terminal blocks must be connected or disconnected with sensor and pre-actuator voltage switched off.

Failure to follow these instructions will result in death or serious injury.

⚠ CAUTION

UNEXPECTED BEHAVIOUR OF APPLICATION

If two 16-pin terminal blocks are used, each can be plugged into the middle or the top connector of the module. Therefore, despite the indicators on the terminal blocks and module, it is possible to invert the two terminal blocks and thus create incorrect wiring.

Plugging the wrong connector could cause unexpected behaviour of the application.

Failure to follow these instructions can result in injury or equipment damage.

The following table shows the procedure for assembling the 10-pin and 16-pin terminal blocks onto a BMX EHC 0200 counting module:

Step	Action
1	Plug the 10-pin terminal block into the bottom connector of the module.
2	Plug the 16-pin terminal block into the middle connector of the module if it is used.
3	Plug the 16-pin terminal block into the top connector of the module if it is used.

NOTE: The three module connectors have indicators which show the proper direction to use for terminal block installation.

How to Connect BMX EHC 0200 Module: Connecting 16-Pin and 10-Pin Terminal Blocks

At a Glance

The BMX EHC 0200 counting module uses the following terminal blocks:

- Two 16-pin terminal blocks for the inputs
- One 10-pin terminal block for supplies outputs

Description of the 10 and 16 Pin Terminal Blocks

The table below shows the characteristics of the BMX EHC 0200 terminal blocks:

<table>
<tr><th colspan="2">Characteristic</th><th>Available</th></tr>
<tr><td colspan="2">Type of terminal block</td><td>Spring terminal blocks</td></tr>
<tr><td colspan="2">Number of wires accommodated</td><td>1</td></tr>
<tr><td rowspan="2">Number of wire gauges accommodated</td><td>minimum</td><td>AWG 24 (0.5 mm^2)</td></tr>
<tr><td>maximum</td><td>AWG 17 (1 mm^2)</td></tr>
<tr><td colspan="2">Wiring constraints</td><td>To insert and remove wires from the connectors, use a 2.5 x 0.4 mm screwdriver to open the round receptacle by pushing on the corresponding plate. Push the flexible plate down on the outside (the side closest to the corresponding receptacle).
A screwing (rotating) or bending motion is not required.</td></tr>
</table>

⚠ DANGER

ELECTRICAL SHOCK

Terminal blocks must be connected or disconnected with sensor and pre-actuator voltage switched off.

Failure to follow these instructions will result in death or serious injury.

BMX EHC 0200 Counting Module Hardware Implementation

5

Subject of this Chapter

This chapter deals with the hardware characteristics of the BMX EHC 0200 module.

What Is in This Chapter?

This chapter contains the following topics:

Characteristics for the BMX EHC 0200 Module and its Inputs and Outputs

General Characteristics

This table presents the general characteristics for the **BMX EHC 0200** and BMX EHC 0200H *(see page 19)* modules:

Module type		2 counting channels	
Maximum frequency at counting inputs		60 kHz	
Number of inputs/outputs per counting channel	Inputs	6 Type three 24 VDC inputs	
	Outputs	Two 24 VDC outputs	
Power Supply	Sensor supply voltage	19.2...30 VDC	
	Module consumption	Does not take into account sensors or encoder consumption ● All inputs OFF: Typical: 15mA ● All inputs ON: Typical: 75mA	
	Actuator supply current	500 mA maximum per output 2 A per module	
Power distribution to sensors		Yes with short-circuit and overload protection - typical 300 mA (short-circuit limited to 2.5 A)	
Hot replacement		Yes, under the following conditions: The module may be removed and reinserted into its location while the rack is switched on, but the counter may have to be revalidated when it is reinserted into its base.	
Dimensions	Width	Module only	32 mm
		On the rack	32 mm
	Height	Module only	103.76 mm
		On the rack	103.76 mm
	Depth	Module only	92 mm
		On the rack	104.5 mm
Encoder compliance		10...30 VDC incremental encoder model with push-pull at outputs	
Insulation voltage of the ground to the bus		1500 V RMS for 1 min	
Rack 24 V supply bus	Current for the 24 V bus	Typical: 40 mA	
Rack 3 V supply bus	Current for the 3 V bus	Typical: 200 mA	
Module Cycle Time		1 ms	

⚠ WARNING

OVERHEATING MODULE

Do not operate the **BMX EHC 0200H** at 70° C (158° F) if the sensor power supply is greater than 26.4 V or less than 21.1 V.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

Input Characteristics

This table presents the general characteristics of the input channels for the module:

Number of inputs per channel			Six 24 VDC inputs
Inputs: IN A, IN B, IN SYNC, IN EN, IN REF, IN CAP	Voltage		30 VDC maximum
	At state 1	Voltage	11 VDC... 30 VDC
		Current	5 mA (up to 30 VDC)
	At state 0	Voltage	< 5 VDC
		Current	< 1.5 mA
	Current at 11 VDC		> 2 mA

Characteristics of Outputs

This table presents the general characteristics of the output channels for the module:

Number of outputs per channel		2
Type		source 24 VDC 0.5 A
Voltage		19.2...30 VCC
Minimum load current		None
Maximum load current	Each point	0.5 A
	Per module	2 A
Leakage current at state 0		0.1 mA maximum
Voltage drop at state 1		3 VDC maximum
Output current short-circuit	Each point	1.5 A maximum
Maximum load capacity		50 µF
Short-circuit and overload		Channel protection

Polarity for each output channel	By default	Normal logic on both channels
	User configuration	Reverse logic for one or several channels
Maximum inductive load		The inductive load is calculated using the following formula: $L = 0{,}5 / I^2 \times F$ The formula above uses the following parameters: • **L:** load inductance in Henry • **I:** load current in Amperes • **F:** switching frequency in Hertz

Display and Diagnostics of the BMX EHC 0200 Counting Module

At a Glance

The BMX EHC 0200 counting module has LEDs that enable the status of the module to be viewed:

- Module state LEDs: RUN, ERR, I/O
- State LEDs for inputs/outputs of each channel: IA, IB, IS, IE, IP, IC, Q0 and Q1.

Illustration

The following drawing shows the display screen of the BMX EHC 0200 module:

Fault Diagnostics

The following table presents the various module states according to the LED states:

Module status	LED indicators										
	ERR	RUN	IO	IA	IB	IS	IE	IP	IC	Q0	Q1
The module is faulty or switched off	○										
The module has a fault	●	○									
The module is not configured	⊘	○	○								
The module has lost communication	⊘	●									
The sensors have a supply fault	○	●	●	⊗							
The actuators have a supply fault	○	●	●							⊗	
Short circuit on output Q0	○	●	●							⊘	
Short circuit on output Q1	○	●	●								⊘
The channels are operational	○	●	○								
The voltage is present at output Q0	○	●	○							●	
The voltage is present at output Q1	○	●	○								●

The voltage is present at input IN_A	○	●	○	●						
The voltage is present at input IN_B	○	●	○		●					
The voltage is present at input IN_SYNC	○	●	○			●				
The voltage is present at input IN_EN	○	●	○				●			
The voltage is present at input IN_REF	○	●	○					●		
The voltage is present at input IN_CAP	○	●	○						●	
Legend										
● LED on										
○ LED off										
⊗ LED flashing slowly										
◍ LED flashing fast										
An empty cell indicates that the state of the LED(s) is not taken into account										

BMX EHC 0200 Module Wiring

At a Glance

The BMX EHC 0200 counting module uses the following:
- Two 16-pin connectors for the inputs
- One 10-pin connector for the outputs

⚠ DANGER

HAZARD OF ELECTRIC SHOCK

- disconnect voltage supplying sensors and pre-actuators before plugging / unplugging the terminal block on the module.
- remove the terminal block before plugging / unplugging the module on the rack.

Failure to follow these instructions will result in death or serious injury.

NOTE: The two 16-pin connectors and the 10-pin connector are sold separately and are available in the BMX XTS HSC 20 connection kit.

Field sensors

The module has type 3 of CEI 1131 inputs that support signals from mechanical switching equipment such as:
- Contact relays
- Push-buttons
- Limit switch sensors
- Switches with 2 or 3 wires

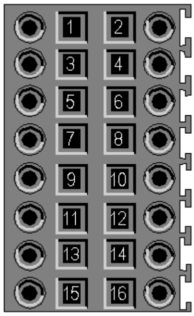
The equipment must have the following characteristics:
- Voltage drop less than 8 V
- Minimum operating current less than or equal to 2 mA
- Maximum current in blocked state less than or equal to 1.5 mA

The module complies with most encoders that have a supply of between 10 and 30 V and push-pull outputs.

NOTE: The module’s 24 V supply for sensors has thermal and short-circuit protection.

Assignment of the 16-Pin Connector

The figure below shows the physical location of the pin numbers for the 16-pin connector:

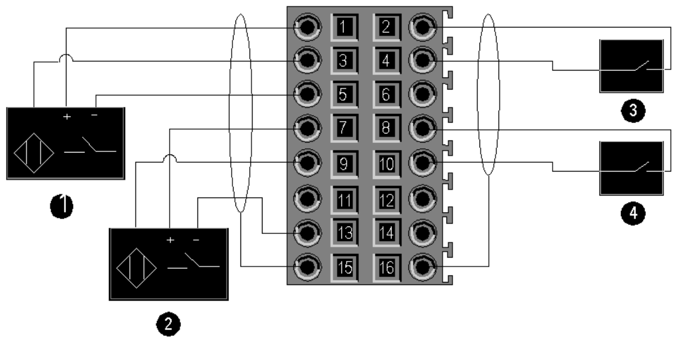


The symbol and description of each pin are described in the table below:

Pin number	Symbol	Description
1, 2, 7, 8	24V_SEN	24 VDC output for sensors supply
5, 6, 13, 14	GND_SEN	24 VDC output for sensors supply
15, 16	FE	Functional earth
3	IN_A	Input A
4	IN_SYNC	Synchronization input
9	IN_B	Input B
10	IN_EN	Enable input selected
11	IN_REF	Homing input
12	IN_CAP	Capture input

Sensor Connections

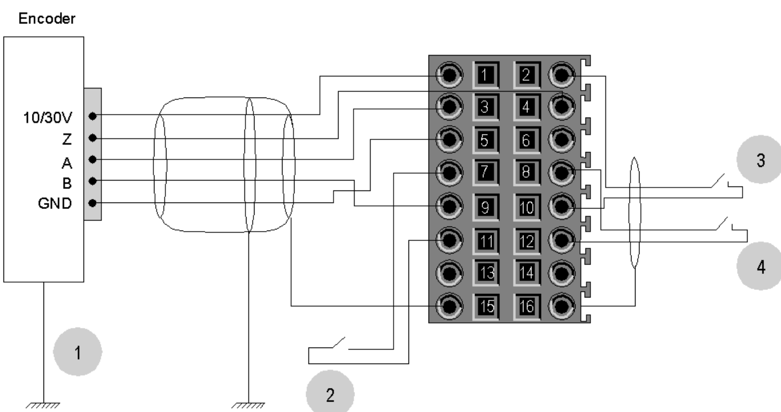
The example below shows sensors with applied to inputs IN_A and IN_B and equipment with applied to inputs IN_EN and IN_SYNC:



1 IN_A input
2 IN_B input
3 IN_SYNC input (synchronization input)
4 IN_EN input (enable input)

Encoder Connection

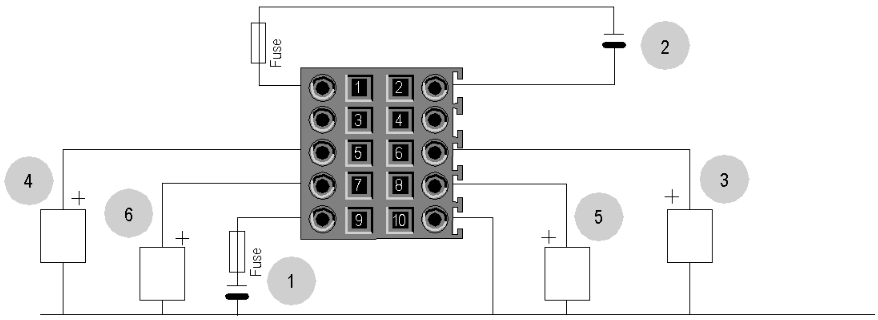
The example below shows an incremental encoder used for axis control and the three auxiliary inputs used especially for the 32-bit counter mode:



1 Encoder (inputs A, B and Z)
2 IN_REF input (homing input)
3 IN_EN input (enable input)
4 IN_CAP input (capture input)

Connecting Outputs and Output Supplies

The figure below shows the connection of supplies and actuators to the 10-pin connector:



1 24 V supply for actuators
2 24 V supply for sensors
3 Actuator for the Q0 output of counting channel 0
4 Actuator for the Q1 output of counting channel 0
5 Actuator for the Q0 output of counting channel 1
6 Actuator for the Q1 output of counting channel 1

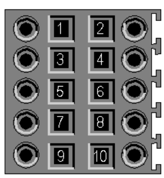
Field Actuators

The Q0 and Q1 outputs are limited by a maximum current of 0.5 A.

NOTE: The Q0 and Q1 outputs have a thermal protection as well as short-circuit protection.

Assignment of the 10-Pin Connector

The figure below shows the physical location of the pin numbers for the 10-pin connector:



The symbol and description of each pin are described in the table below:

Pin number	Symbol	Description
1	24V_IN	24 VDC input for sensors supply
2	GND_IN	24 VDC input for sensors supply
5	Q0-1	Q1 output for counting channel 0
6	Q0-0	Q0 output for counting channel 0
7	Q1-1	Q1 output for counting channel 1
8	Q1-0	Q0 output for counting channel 1
9	24V_OUT	24 VDC input for actuators supply
10	GND_OUT	24 VDC input for actuators supply

Safety Instructions

WARNING

UNEXPECTED EQUIPMENT OPERATION

Follow those instructions to reduce electromagnetic perturbations:

- adapt the programmable filtering to the frequency applied at the inputs, or
- use a shielded cable (connected to the functional ground) connected to pins 15 and 16 of the connector when using an encoder or a fast detector.

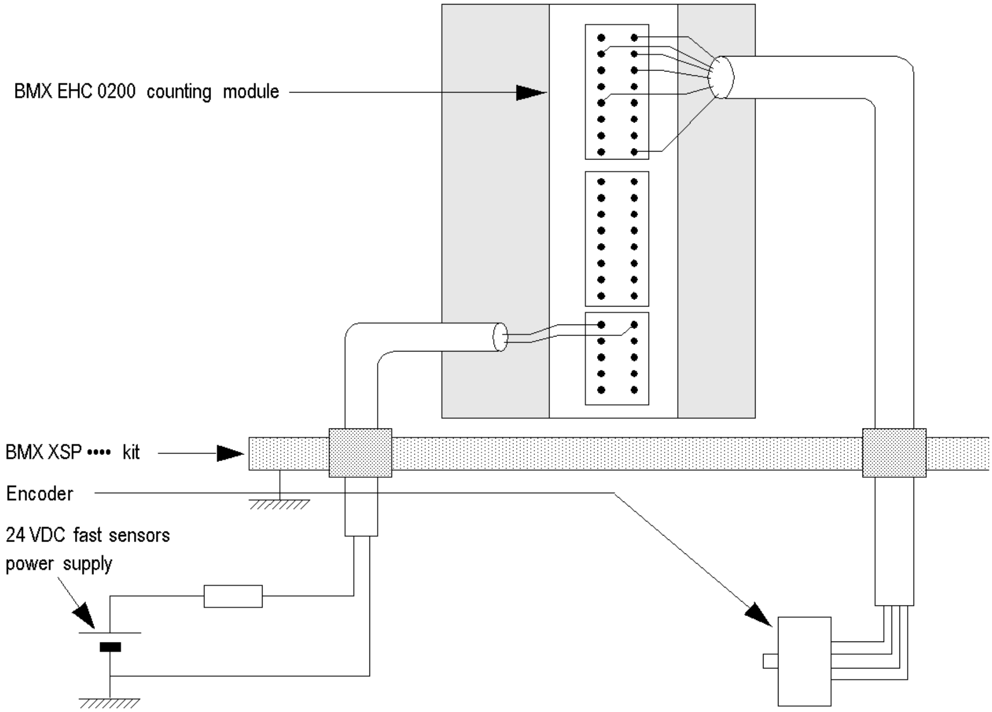
In a highly disturbed environment,

- use the BMX XSP 0400/0600/0800/1200 electromagnetic protection kit *(see Modicon M340 Using Unity Pro, Processors, Racks, and Power Supply Modules, Setup Manual)* (See Modicon M340 using Unity Pro, Processors, Racks and Power Supply Modules, BMX XSP xxx Protection Bar) to connect the shielding without programmable filtering and
- use a specific 24 VDC supply for inputs and a shielded cable for connecting the supply to the module.

Electromagnetic perturbations may cause the application to operate in an unexpected manner.

Failure to follow these instructions can result in death, serious injury, or equipment damage.

The figure below shows the recommended circuit for high-noise environment using the BMX XSP 0400/0600/0800/1200 electromagnetic protection kit:



⚠ CAUTION

POTENTIAL MODULE DAMAGE - IMPROPER FUSE SELECTION

Use fast acting fuses to protect the electronic components of the module from overcurrent and reverse polarity of the input/output supplies. Improper fuse selection could result to damage to the module.

Failure to follow these instructions can result in injury or equipment damage.

Counting Module BMX EHC 0200 Functionalities

III

BMX EHC 0200 Counting Module Functionalities

6

Subject of this Chapter

This chapter deals with functionalities and counting modes of the BMX EHC 0200 module.

What Is in This Chapter?

This chapter contains the following sections:

6.1 BMX EHC 0200 Module Configuration

Subject of this Section

This section deals with the configuration of the BMX EHC 0200 module.

What Is in This Section?

This section contains the following topics:

Input Interface Blocks

Description

The BMX EHC 0200 counting module has six inputs:

- 3 fast inputs
- 3 classic inputs

Fast Inputs

The table below presents the module's fast inputs.

Input	Use with sensors	Use with an encoder
IN_A input	Clock input for measurement or single upcounting	For signal A
IN_B input	Second clock input for differential counting or measurement	For signal B
IN_SYNC input	Main synchronization input used for starting and homing	For signal Z Used for homing

Classic Inputs

The table below presents the module's classic inputs:

Input	Use
IN_EN input	Used to authorize counter operation
IN_REF input	Used for homing in advanced mode
IN_CAP input	Used for register capture

Programmable Filtering

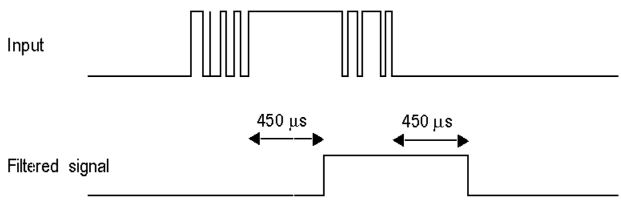
At a Glance

The BMX EHC 0200 counting module's six inputs are compatible with the use of mechanical switches.

A programmable debounce filter with 3 levels (low, medium and high) is available at every input.

Debounce Filter Diagram

The figure below shows the debounce filter with a low filtering level:



In this mode, the system delays all transitions until the signal is stable for 450 µs.

Selecting the Filtering Level

The table below specifies the characteristics of each input for the selected level of filtering:

Filtering level	Input	Maximum delay	Minimum pulse	Maximum frequency
None	IN_A, IN_B	-	5 µs	60 KHz
	IN_SYNC	-	5 µs	200 Hz
	IN_EN	50 µs	-	-
	IN_CAP, IN_REF	-	50 µs	200 Hz
Low for bounces > 2 KHz	IN_A, IN_B	-	450 µs	1 KHz
	IN_EN	450 µs	-	-
	IN_SYNC, IN_CAP, IN_REF	-	500 µs	200 Hz
Resource for bounces > 1 KHz	IN_A, IN_B	-	1.25 ms	350 Hz
	IN_EN	1.25 ms	-	-
	IN_SYNC, IN_CAP, IN_REF	-	1.25 ms	200 Hz
High for bounces > 250 Hz	IN_A, IN_B	-	4.2 ms	100 Hz
	IN_EN	4.2 ms	-	-
	IN_SYNC, IN_CAP, IN_REF	-	4.2 ms	100 Hz

Comparison

At a Glance

The comparison block operates automatically. This block is available in certain counting modes:

- Frequency
- Period measuring
- Ratio
- One shot counter
- Modulo loop counter
- Free large counter

Comparison Thresholds

The comparison block has two thresholds:

- The upper threshold: `upper_th_value` double word (`%QDr.m.c.4`)
- The lower threshold: `lower_th_value` double word (`%QDr.m.c.2`)

The upper threshold value must be greater than the lower threshold value.

If the upper threshold is less than or equal to the lower threshold, the lower threshold does not change but it is ignored.

This rule takes into account the format of the counter value.

Comparison Status Register

The result of the comparison is stored in the `compare_status` register (`%IWr.m.c.1`).

The values of the two capture registers and the current value of the counter are compared with the thresholds.

The possible results are:

- Low: The value is less than the lower threshold value.
- Window: The value is between the upper and lower thresholds or equal to one of the two thresholds.
- High: The value is greater than the upper threshold.

The `compare_enable`register (`%IWr.m.c.1`) consists of:

Status register bit	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Compared element								Capture 1			Capture 0			Counter		
Comparison result								High	Window	Low	High	Window	Low	High	Window	Low

Update

When the `compare_enable` bit (`%QWr.m.c.0.5`) is set to 0, the comparison status register is deleted.

The comparison with capture 0 and capture 1 registers values is performed every time the registers are loaded.

The comparison with the counter current value is performed as follows:

Counting mode	Registers updating
Frequency	Intervals of 10 ms
Period measuring	At the end of the period
Ratio	Intervals of 10 ms
Event counting	Period intervals defined by the user
One shot counter	Intervals of 1 ms Counter reloading Counter stops Threshold crossing
Modulo loop	Intervals of 1 ms Counter reloading or resetting to 0 Counter stops Threshold crossing
Free large counter	Intervals of 1 ms Counter reloading Threshold crossing
Pulse width modulation	Function not available in this mode

Modification of the Thresholds during the Operational Phase

When the `compare_enable` bit (`%QWr.m.c.0.5`) is set to 0, the comparison status register is deleted.

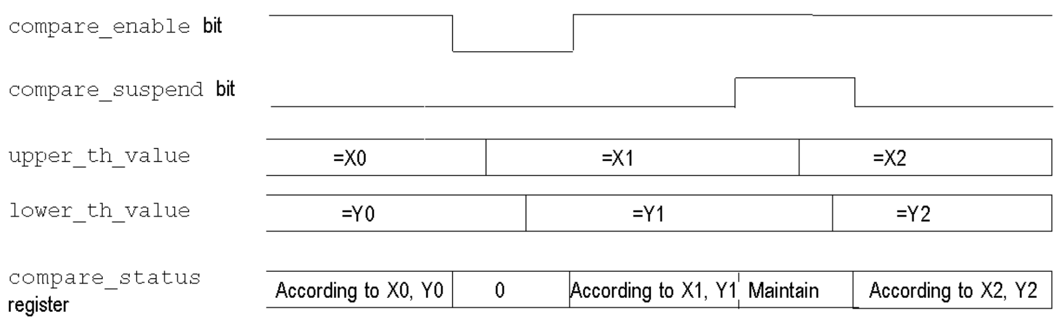
When the `compare_suspend` bit (`%QWr.m.c.0.6`) is set to 1, the value of the comparison status register is frozen until the bit switches back to 0.

The application may change threshold values without causing any disturbance when the `compare_suspend` bit (`%QWr.m.c.0.6`) is set to 1.

This functionality allows modifying the application thresholds without modifying the status register behaviour.

When this bit switches back to 0, the comparisons restart with new threshold values.

The following figure illustrates the actions of the `compare_enable` bit (`%QWr.m.c.0.5`) and the `compare_suspend` bit (`%QWr.m.c.0.6`):



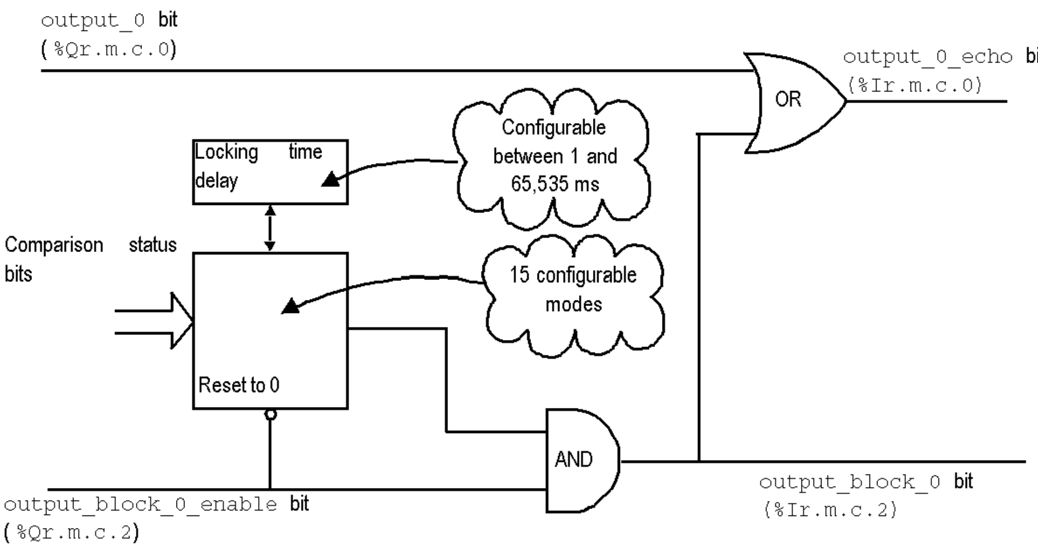
Output Block Functions

Output Function Blocks

Every channel in the counting module has two programmable output blocks that operate with the comparison status register and affect the behavior of physical outputs Q0 and Q1.

There are two ways to control the output:
- From the application: in this case, the output corresponds to the status of the output bit from the output command bit.
- From the output function block: in this case, the user must enable the output block function. Then, the output corresponds to the status of the output bit from the function block.

The following figure shows the output function block Q0:



Use of the Function Block

Every physical output is controlled by two bits:
- output_block_0_enable (%Qr.m.c.2) and output_0 (%Qr.m.c.0) for block 0
- output_block_1_enable (%Qr.m.c.3) and output_1 (%Qr.m.c.1) for block 1

The output_block_0(1)_enable bit enables the operation of the function block 0(1) to be authorized when it is set to 1. When the bit is set to 0, Bit output_block_0(1) is maintained at 0.

The `output_0(1)` bit is applied at the logic output Q0(1) and must be set to 0 when the function block is used. When the bit is set to 1, the output is forced to 1.

In the operational modes where the block generates a pulse, the pulse width can be configured thanks to the configuration screen.

Output Programming

The table below shows the configurable functions:

Function code	Programming
0	Disabled = no direct action (Default value)
1	Low counter. The output is high if the counter value is less than the low threshold.
2	Counter in a window The output is high if the counter value is between the upper and lower thresholds or equal to one of the two thresholds.
3	High counter. The output is high if the counter value is greater than the upper threshold.
4	Pulse less than the lower threshold. The output pulse starts when the counter value decreases and crosses the lower threshold value -1.
5	Pulse greater than the lower threshold. The output pulse starts when the counter value increases and crosses the lower threshold value +1.
6	Pulse less than the upper threshold. The output pulse starts when the counter value decreases and crosses the upper threshold value -1.
7	Pulse greater than the upper threshold. The output pulse starts when the counter value increases and crosses the upper threshold value +1.
8	Counter stopped (only in one shot counter mode). The output changes to high if the counter is stopped.
9	Counter running (only in one shot counter mode). The output changes to high if the counter is running.
10	Capture 0 low value. The output is high if the capture 0 value is less than the lower threshold.
11	Capture 0 value in a window. The output is high if the capture 0 value is between the upper and lower thresholds or equal to one of the two thresholds.
12	Capture 0 high value. The output is high if the capture 0 value is greater than the upper threshold.

Function code	Programming
13	Capture1 low value. The output is high if the capture1 value is less than the lower threshold.
14	Capture1 value in a window. The output is high if the capture1 value is between the upper and lower thresholds or equal to one of the two thresholds.
15	Capture1 high value. The output is high if the capture1 value is greater than the upper threshold.

NOTE: The output 0 function block is inactive when using the counter in pulse width modulation mode.

Output Performances

In general, these reflex actions act with a delay less than 0.6 ms. The repeatability is about +/- 0.3 ms.

Special boost functions:
- "Counter Low" (function code 1) applied to Output Block 0
- "Counter High" (function code 3) applied to Output Block 1 speed up timing.

Delay is less than 0.2 ms. The repeatability is about +/- 1 s.

Output Properties

The counting module BMX EHC 0200 enables output signals to be exchanged with two 24VCC field actuators.

It is possible to configure the following parameters for each output:
- The module response for fault recovery
- The output polarity for each counting channel (positive or negative polarity)
- The fallback mode and state for every module channel

These three parameters are described in the following pages.

Fault Recovery response

Outputs Q0 and Q1 are current limited (0.5 A maximum).

A thermal shutdown protects each output.

When a short-circuit is detected on one of the output channels, the counting module enables one of the two following actions according to the configuration:

- `fault recovery` parameter configured as `latched off`: The counting module latches off the output channel
- `fault recovery` parameter configured as `autorecovery`: The counting module latches off the output channel and automatically attempts to recover the error and to resume operation on the channel when the error is corrected.

In case of the `fault recovery` parameter is configured to `latched off`, if an output channel has been latched off because of short-circuit detection, the counting module recovers the fault upon the following sequence is processed:

- The error has been corrected
- You explicitly reset the fault: To reset the error, the application software must:
  - Reset the `output_block_enable` bit if it is active
  - Command the ouput to 0 V (depends on the polarity).

In case of the `fault recovery` parameter is configured to `auto recovery`, an output channel that has been turned off because of error detection starts operating again as soon as the error is corrected. No user intervention is required to reset the channels.

NOTE: A minimum delay of 10 s occurs before the error is cleared in both latched off and auto recovery modes.

Output Polarity Programming

It is possible to configure the `polarity` parameter for each output during the channel configuration:

- `polarity` parameter configured as `polarity +`: The physical output is 24 VDC when the output is at the high level (`output_0_echo` = 1)
- `polarity` parameter configured as `polarity -`: The physical output is 24 VDC when the output is at the low level (`output_0_echo` = 0)

By default, the two output channels are in positive polarity.

Output Fallback Modes

The fallback modes are the predefined states to which the output channels revert when the channel is not controlled by the processor (when communications are lost or when the processor is stopped for example).

The fallback mode of each output channel can be configured as one of the following modes:

- Fallback value: With. You may configure the fallback value to apply as 0 or 1
- Fallback value: Without. The output block function continues to operate according to the last received commands.

NOTE: By default, the fallback mode of the 2 output channels is `with` and the `fallback value` parameter is `0`.

Diagnostics

Consistency Rules for Inputs Interface

The input interface requires that the sensor power supply remains active for counting operations.

When the sensor power supply interrupts lasts 1 ms or less, the counter remains stable.

In case of power interrupt is greater than 1 ms, all counter values are disabled.

By default, the sensor supply fault makes the `CH_ERROR` (`%Ir.m.c.ERR`) global status bit at the high level and the red led IO lighted.

The configuration screen allows to unlink the sensor supply fault to the `CH_ERROR` bit by configuring the parameter `Input Supply Fault` as `local` instead of `General IO Fault`.

`IODDT_VAR1` is of the type `T_Unsigned_CPT_BMX` or `T_Signed_CPT_BMX`

Consistency Rules for Outputs Interface

The output interface requires that the actuator power supply remains active for output blocks functions operations.

When the actuator supply voltage is insufficient the ouputs are held to 0 V.

By default, the actuator supply fault makes the `CH_ERROR` (`%Ir.m.c.ERR`) global status bit at the high level and the red led IO lighted.

The configuration screen allows to unlink the actuator supply fault to the `CH_ERROR` bit by configuring the parameter `Output Supply Fault` as `local` instead of `General IO Fault`.

`IODDT_VAR1` is of the type `T_Unsigned_CPT_BMX` or `T_Signed_CPT_BMX`

Explicit channel status words

The table below presents the composition of the `%MWr.m.c.2` and `%MWr.m.c.3` status words:

Status Word	Bit position	Designation
`%MWr.m.c.2`	0	External fault at inputs
	1	External fault at outputs
	4	Internal error or self-testing.
	5	Configuration Fault
	6	Communication Error
	7	Application fault

Status Word	Bit position	Designation
%MWr.m.c.3	2	Sensor supply fault
	3	Actuator supply fault
	4	Short circuit on output Q0
	5	Short circuit on output Q1

IO Data

All input/output statuses are provided in the channel data bits.

The table below shows the channel data bits:

Input/Output data field	Designation
%Ir.m.c.0	Logical state of output Q0
%Ir.m.c.1	Logical state of output Q1
%Ir.m.c.2	State of the output block function 0
%Ir.m.c.3	State of the output block function 1
%Ir.m.c.4	Electrical state of IN_A input
%Ir.m.c.5	Electrical state of IN_B input
%Ir.m.c.6	Electrical state of IN_SYNC input
%Ir.m.c.7	Electrical state of IN_EN input
%Ir.m.c.8	Electrical state of IN_REF input
%Ir.m.c.9	Electrical state of IN_CAP input

Synchronization, Homing, Enable, Reset to 0 and Capture Functions

Introduction

This section presents the functions used by the various counting modes of the BMX EHC 0200 module:

- Synchronization function
- Homing function
- Enable function
- Reset to 0 function
- Capture functions

Each function uses at least one of the following two bits:

- `valid_(function)` bit: Setting this bit to 1 allows you to take into account the occurrence of an external event which activates the function. If this bit is set to 0, the event is not taken into account and does not activate the function. The `functions_enabling` word (`%QWr.m.c.0`) contains all the `valid_(function)` bits.
- `force_(function)` bit: Setting this bit to 1 allows you to activate the function irrespective of the status of the external event. All the `force_(function)` bits are `%Qr.m.c.4`...`%Qr.m.c.8` language objects.

Synchronization Function

The synchronization function is used to synchronize the counter operation upon a transition applied to the IN_SYNC (%I r.m.c.6) physical input or the `force_sync` bit set to 1.

This function is usable in the following counting modes:

- Pulse width modulation: to restart the output signal at the beginning (phase at 1)
- Modulo loop counter: to reset and start the counter
- One shot counter: to preset and start the counter
- Event counting: to restart the internal time base at the beginning

The user may configure the `synchro edge` parameter in the configuration screen by choosing from the following two possibilities to configure the sensitive edge that carries out the synchronization:

- Rising edge of the IN_SYNC input
- Falling edge of the IN_SYNC input

The following table presents the `force_sync` bit in bold which is an element of the `%Qr.m.c.d` output command word:

Language object	Standard symbol	Meaning
%Qr.m.c.0	OUTPUT_0	Forces OUTPUT_0 to level 1
%Qr.m.c.1	OUTPUT_1	Forces OUTPUT_1 to level 1
%Qr.m.c.2	OUTPUT_BLOCK_0_ENABLE	Implementation of output 0 function block
%Qr.m.c.3	OUTPUT_BLOCK_1_ENABLE	Implementation of output 1 function block
%Qr.m.c.4	**FORCE_SYNC**	**Counting function synchronization and start**
%Qr.m.c.5	FORCE_REF	Set to preset counter value
%Qr.m.c.6	FORCE_ENABLE	Implementation of counter
%Qr.m.c.7	FORCE_RESET	Reset counter
%Qr.m.c.8	SYNC_RESET	Reset SYNC_REF_FLAG
%Qr.m.c.9	MODULO_RESET	Reset MODULO_FLAG

The following table presents the `valid_sync` bit in bold which is an element of the `%QWr.m.c.0` function enabling word:

Language object	Standard symbol	Meaning
%QWr.m.c.0.0	**VALID_SYNC**	**Synchronization and start authorization for the counting function via the IN_SYNC input**
%QWr.m.c.0.1	VALID_REF	Operation authorization for the internal preset function
%QWr.m.c.0.2	VALID_ENABLE	Authorization of the counter enable via the IN_EN input
%QWr.m.c.0.3	VALID_CAPT_0	Capture authorization in the capture0 register
%QWr.m.c.0.4	VALID_CAPT_1	Capture authorization in the capture1 register
%QWr.m.c.0.5	COMPARE_ENABLE	Comparators operation authorization
%QWr.m.c.0.6	COMPARE_SUSPEND	Comparator frozen at its last value

The following table presents the synchronization principle:

Edge	**Status of the `valid_sync` (`%QWr.m.c.0.0`) bit**	**Status of the counter**
Rising or falling edge on IN_SYNC (depending on the configuration)	Set to 0	Not synchronized
Rising or falling edge on IN_SYNC (depending on the configuration)	Set to 1	Synchronized
Rising edge on `force_sync` (`%Qr.m.c.4`) bit	Set to 0 or 1	Synchronized

When the synchronization occurs, the application can react using :

- either the SYNC_REF_FLAG input (%IWr.m.c.0.2) *(see page 68)*
- or the EVT_SYNC_PRESET input (%IWr.m.c.10.2) *(see page 70)*.

Homing Function

This homing function loads the value predefined in the adjust screen `preset value` (`%MDr.m.c.6`) into the counter when the preset condition (defined by the `preset mode` parameter) occurs. This preset condition takes into account the IN_SYNC and IN_REF physical inputs to define the reference point of the process.

This function is only used in the free large counter mode.

The user may change the `Preset Mode` parameter in the configuration screen by choosing from the following five possibilities to configure the preset condition:

- Rising edge of the IN_SYNC input
- Rising edge of the IN_REF input
- Rising edge of the IN_SYNC input and high level of the IN_REF input
- First rising edge of the IN_SYNC input and high level of the IN_REF input
- First rising edge of the IN_SYNC input and low level of the IN_REF input

The following table presents the `force_ref` bit in bold which is an element of the `%Qr.m.c.d` output command word:

Language object	**Standard symbol**	**Meaning**
`%Qr.m.c.0`	OUTPUT_0	Forces OUTPUT_0 to level 1
`%Qr.m.c.1`	OUTPUT_1	Forces OUTPUT_1 to level 1
`%Qr.m.c.2`	OUTPUT_BLOCK_0_ENABLE	Implementation of output 0 function block
`%Qr.m.c.3`	OUTPUT_BLOCK_1_ENABLE	Implementation of output 1 function block

Language object	Standard symbol	Meaning
%Qr.m.c.4	FORCE_SYNC	Counting function synchronization and start
%Qr.m.c.5	**FORCE_REF**	**Set to preset counter value**
%Qr.m.c.6	FORCE_ENABLE	Implementation of counter
%Qr.m.c.7	FORCE_RESET	Reset counter
%Qr.m.c.8	SYNC_RESET	Reset SYNC_REF_FLAG
%Qr.m.c.9	MODULO_RESET	Reset MODULO_FLAG

The following table presents the valid_ref bit in bold which is an element of the %QWr.m.c.0 function enabling word:

Language object	Standard symbol	Meaning
%QWr.m.c.0.0	VALID_SYNC	Synchronization and start authorization for the counting function via the IN_SYNC input
%QWr.m.c.0.1	**VALID_REF**	**Operation authorization for the internal preset function**
%QWr.m.c.0.2	VALID_ENABLE	Authorization of the counter enable via the IN_EN input
%QWr.m.c.0.3	VALID_CAPT_0	Capture authorization in the capture0 register
%QWr.m.c.0.4	VALID_CAPT_1	Capture authorization in the capture1 register
%QWr.m.c.0.5	COMPARE_ENABLE	Comparators operation authorization
%QWr.m.c.0.6	COMPARE_SUSPEND	Comparator frozen at its last value

The following table presents the homing principle:

Edge	Status of the valid_ref bit (%QWr.m.c.0.1)	Status of the counter
Homing condition edge (depending on the configuration)	Set to 0	Not preset
Homing condition edge (depending on the configuration)	Set to 1	Preset
Rising edge on force_ref bit (%Qr.m.c.5)	Set to 0 or 1	Preset

When the preset occurs consequently to the preset condition, the application can react using:

- either the SYNC_REF_FLAG input (%IWr.m.c.0.2) *(see page 68)*
- or the EVT_SYNC_PRESET input (%IWr.m.c.10.2) *(see page 70)*.

Enable Function

This function is used to authorize changes to the current counter value depending on the status of the IN_EN physical input.

This function is used in the following counting modes:

- Pulse width modulation
- Modulo loop counter
- One shot counter
- Free large counter

The following table presents the `force_enable` bit in bold which is an element of the `%Qr.m.c.d` output command word:

Language object	Standard symbol	Meaning
%Qr.m.c.0	OUTPUT_0	Forces OUTPUT_0 to level 1
%Qr.m.c.1	OUTPUT_1	Forces OUTPUT_1 to level 1
%Qr.m.c.2	OUTPUT_BLOCK_0_ENABLE	Implementation of output 0 function block
%Qr.m.c.3	OUTPUT_BLOCK_1_ENABLE	Implementation of output 1 function block
%Qr.m.c.4	FORCE_SYNC	Counting function synchronization and start
%Qr.m.c.5	FORCE_REF	Set to preset counter value
%Qr.m.c.6	**FORCE_ENABLE**	**Implementation of counter**
%Qr.m.c.7	FORCE_RESET	Reset counter
%Qr.m.c.8	SYNC_RESET	Reset SYNC_REF_FLAG
%Qr.m.c.9	MODULO_RESET	Reset MODULO_FLAG

The following table presents the `valid_enable` bit in bold which is an element of the `%QWr.m.c.0` function enabling word:

Language object	Standard symbol	Meaning
%QWr.m.c.0.0	VALID_SYNC	Synchronization and start authorization for the counting function via the IN_SYNC input
%QWr.m.c.0.1	VALID_REF	Operation authorization for the internal preset function

Language object	Standard symbol	Meaning
%QWr.m.c.0.2	**VALID_ENABLE**	**Authorization of the counter enable via the IN_EN input**
%QWr.m.c.0.3	VALID_CAPT_0	Capture authorization in the capture0 register
%QWr.m.c.0.4	VALID_CAPT_1	Capture authorization in the capture1 register
%QWr.m.c.0.5	COMPARE_ENABLE	Comparators operation authorization
%QWr.m.c.0.6	COMPARE_SUSPEND	Comparator frozen at its last value

The following table presents the validation principle:

Condition	Status of the `valid_enable` bit (`%QWr.m.c.0.2`) and `force_enable` bit (`%Qr.m.c.6`)	Status of the counter
IN_EN set to 1	The 2 bits are set to 0	Not counting (frozen)
IN_EN set to 1	At least one of the two bits is set to 1	Counting (free)

Reset to 0 Function

This function is used to load the value 0 into the counter via software command.

This function is used in the following counting modes:

- Free large counter
- Modulo loop counter
- One shot counter

The following table presents the `force_reset` bit in bold which is an element of the `%Qr.m.c.d` output command word:

Language object	Standard symbol	Meaning
%Qr.m.c.0	OUTPUT_0	Forces OUTPUT_0 to level 1
%Qr.m.c.1	OUTPUT_1	Forces OUTPUT_1 to level 1
%Qr.m.c.2	OUTPUT_BLOCK_0_ENABLE	Implementation of output 0 function block
%Qr.m.c.3	OUTPUT_BLOCK_1_ENABLE	Implementation of output 1 function block
%Qr.m.c.4	FORCE_SYNC	Counting function synchronization and start
%Qr.m.c.5	FORCE_REF	Set to preset counter value

Language object	Standard symbol	Meaning
%Qr.m.c.6	FORCE_ENABLE	Implementation of counter
%Qr.m.c.7	**FORCE_RESET**	**Reset counter**
%Qr.m.c.8	SYNC_RESET	Reset SYNC_REF_FLAG
%Qr.m.c.9	MODULO_RESET	Reset MODULO_FLAG

The function is only activated by the rising edge of the `force_reset` bit (`%Qr.m.c.7`). There is no `valid_reset` bit because the function is not activated by any physical input.

Capture Function

This function allows to store the current counter value into a capture register upon an external condition.

Each BMX EHC 0200 module channel has 2 capture registers:

- `capture0`
- `capture1`.

The capture function is used in the following counting modes:

- Modulo loop counter
- Free large counter.

In the modulo loop counter mode, only the `capture0` function is available.

The function enables to record the current counter value according to the synchronisation condition.

If the IN_SYNC input receives the sensitive edge of synchronization *(see page 60)*, the current counter value is stored into the `capt_0_val` register (`%IDr.m.c.14`). The `valid_capt_0` bit (`%QWr.m.c.0.3`) must be set to 1 to operate.

When the synchronization is requiered at the same time (with the `valid_sync` bit set to 1) the storage into the `capt_0_val` register occurs just before reseting the current counter value.

In the free large counter mode, both `capture0` and `capture1` registers are available.

The `capture1` function always stores the current counter value into the `capt_1_val` register (`%IDr.m.c.16`) as soon as the IN_CAP input receives a rising edge. The `valid_capt_1` bit (`%QWr.m.c.0.4`) must be set to 1 to operate.

The capture0 function can be configured as one of the following 2 conditions:

- Preset condition
- Falling edge of the IN_CAP input.

The valid_capt_0 bit (%QWr.m.c.0.3) must be set to 1 to operate.

If the capture0 function is configured as the preset condition, the function stores the current counter value into the capt_0_val register (%IDr.m.c4) when the defined preset condition *(see page 62)* occurs.

When the preset is requiered at the same time (with the valid_ref bit set to 1) the storage into the capt_0_val register occurs just before loading the current counter value at the preset value.

In all cases, the current counter value must be valid before the capture event (the validity bit (%IWr.m.c.0.3) set to 1)

Modulo Flag and Synchronization Flag

At a Glance

This section presents the operation of the bits relating to the following events:

- Synchronization or counter homing event, depending on the counting mode.
- Counter rollovers the modulo or its limits in forward or reverse.

The table below presents the counting modes that may activate synchronization, homing and modulo events:

Flag	Counting mode concerned
sync_ref_flag bit (%IWr.m.c.0.2)	• Free Large counter: When the counter presets • Modulo loop counter: When the counter resets • One shot counting: When the counter presets and starts
modulo_flag bit (%IWr.m.c.0.1	• Modulo loop counter: When the counter rollovers the modulo or 0 • Free large counter: When the counter rollovers its limits.

Operation of the Flag Bits

The synchronization or homing event's flag bit is set to 1 when a counter synchronization or homing occurs.

The modulo event's flag bit is set to 1 in the following counting modes:

- Modulo loop counter mode: the flag bit is set to 1 when the counter rollovers the modulo
- Free large counter mode: the flag bit is set to 1 when the counter rollovers its limits in forward or reverse

Location of the Flag Bits

The following table presents the modulo_flag and sync_ref_flag bits which are elements of the %IWr.m.c.d status word:

Language object	Standard symbol	Meaning
%IWr.m.c.0.0	RUN	The counter operates in one shot mode only
%IWr.m.c.0.1	**MODULO_FLAG**	**Flag set to 1 by a modulo switch event**
%IWr.m.c.0.2	**SYNC_REF_FLAG**	**Flag set to 1 by a preset or synchronization event**
%IWr.m.c.0.3	VALIDITY	The current numerical value is valid

Language object	Standard symbol	Meaning
%IWr.m.c.0.4	HIGH_LIMIT	The current numerical value is locked at the upper threshold value
%IWr.m.c.0.5	LOW_LIMIT	The current numerical value is locked at the lower threshold value

Resetting the Flag Bits to 0

The user application must reset the flag bit to 0 (if it is active) by using the appropriate command bit from the following two bits:

- sync_reset (%IWr.m.c.8) bit to reset the synchronization or homing event's flag bit to 0
- modulo_reset (%IWr.m.c.9) bit to reset the modulo reached event's flag bit to 0

Location of Reset to 0 Commands

The following table presents the sync_reset and modulo_reset bits which are elements of the %Qr.m.c.d output command word:

Language object	Standard symbol	Meaning
%Qr.m.c.0	OUTPUT_0	Forces OUTPUT_0 to level 1
%Qr.m.c.1	OUTPUT_1	Forces OUTPUT_1 to level 1
%Qr.m.c.2	OUTPUT_BLOCK_0_ENABLE	Implementation of output 0 function block
%Qr.m.c.3	OUTPUT_BLOCK_1_ENABLE	Implementation of output 1 function block
%Qr.m.c.4	FORCE_SYNC	Counting function synchronization and start
%Qr.m.c.5	FORCE_REF	Set to preset counter value
%Qr.m.c.6	FORCE_ENABLE	Implementation of counter
%Qr.m.c.7	FORCE_RESET	Reset counter
%Qr.m.c.8	**SYNC_RESET**	**Reset SYNC_REF_FLAG**
%Qr.m.c.9	**MODULO_RESET**	**Reset MODULO_FLAG**

Sending Counting Events to the Application

At a Glance

The event task number must be declared in the module's configuration screen to enable the events sending.

The BMX EHC 0200 module has eight event sources contained in the `events_source` word at the address `%IWr.m.c.10`:

Address	Standard Symbol	Description	Counting mode concerned
%IWr.m.c.10.0	EVT_RUN	Event due to start of counting.	One Shot Counter mode
%IWr.m.c.10.1	EVT_MODULO	Event due to counter being equal to modulo value - 1 or equal to value 0.	• Modulo Loop Counter Mode • Free Large Counter mode
%IWr.m.c.10.2	EVT_SYNC_PRESET	Event due to a synchronization or counter homing.	• Event Counting mode • Period Measuring mode • One Shot Counter mode • Modulo Loop Counter mode • Free Large Counter mode
%IWr.m.c.10.3	EVT_COUNTER_LOW	Event due to counter being less than the lower threshold.	• Frequency mode • Event Counting mode • Period Measuring mode • Ratio mode • One Shot Counter mode • Modulo Loop Counter mode • Free Large Counter mode
%IWr.m.c.10.4	EVT_COUNTER_WINDOW	Event due to counter being between the upper and lower thresholds.	• Frequency mode • Event Counting mode • Period Measuring mode • Ratio mode • One Shot Counter mode • Modulo Loop Counter mode • Free Large Counter mode
%IWr.m.c.10.5	EVT_COUNTER_HIGH	Event due to counter being greater than the upper threshold.	• Frequency mode • Event Counting mode • Period Measuring mode • Ratio mode • One Shot Counter mode • Modulo Loop Counter mode • Free Large Counter mode
%IWr.m.c.10.6	EVT_CAPT_0	Event due to capture 0.	• Modulo Loop Counter Mode • Free Large Counter mode

Address	Standard Symbol	Description	Counting mode concerned
%IWr.m.c.10.7	EVT_CAPT_1	Event due to capture 1.	Free Large Counter mode
%IWr.m.c.10.8	EVT_OVERRUN	Event due to overrun	• Frequency mode • Event Counting mode • Period Measuring mode • Ratio mode • One Shot Counter mode • Modulo Loop Counter mode • Free Large Counter mode

All the events sent by the module, whatever their source, call the same single event task in the PLC.

There is normally only one type of event indicated per call.

The `evt_sources` word (`%IWr.m.c.10`) is updated at the start of the event task processing.

Enabling Events

In order for a source to produce an event, the validation bit corresponding to the event must be set to 1:

Address	Description
%QWr.m.c.1.0	Start of counting event validation bit.
%QWr.m.c.1.1	Counter rollovering modulo, 0 or its limits event validation bit.
%QWr.m.c.1.2	Synchronization or counter homing event validation bit.
%QWr.m.c.1.3	Counter less than lower threshold event validation bit.
%QWr.m.c.1.4	Counter between the upper and lower thresholds event validation bit.
%QWr.m.c.1.5	Counter greater than upper threshold event validation bit.
%QWr.m.c.1.6	Capture 0 event validation bit.
%QWr.m.c.1.7	Capture 1 event validation bit.

Input Interface

The event only has one input interface. This interface is only updated at the start of the event task processing. The interface consists of:

- The `evt_sources` word (`%IWr.m.c.10`)
- The current value of the counter during the event (or an approximate value) contained in the `counter_value` word (`%IDr.m.c.12`)
- The `capt_0_val` register (`%IDr.m.c.14`) updated if the event is the capture 0
- The `capt_1_val` register (`%IDr.m.c.16`) updated if the event is the capture 1

Operating Limits

Each counter channel can produce a maximum of one event per millisecond, but this flow may be slowed down by simultaneously sending events to several modules on the PLC bus.

Each counter channel has a four slot transmission buffer which can be used to store several events while waiting to be sent.

If the channel is unable to send all of the internally produced events, the `overrun_evt` bit (`%IWr.m.c.10.8`) of the `evt_sources` word is set to 1.

6.2 BMX EHC 0200 Module Operation Modes

Subject of this Section

This section deals with the different counting modes of the BMX EHC 0200 module.

What Is in This Section?

This section contains the following topics:

BMX EHC 0200 Module Operation in Frequency Mode

At a Glance

Using the frequency mode allows you to measure an event frequency, speed, rate and flow.

Basic Principle

In this mode, the module monitors the pulses applied only to the IN_A input and calculates the number of pulses in time intervals of 1 s. The current frequency is then shown in number of events per second (hertz). The counting register is updated at the end of each 10 ms interval.

Counter Status Bits in Frequency Mode

The table below shows the composition of the counter's `%IWr.m.c.0` status word in frequency mode.

Bit	Label	Description
`%IWr.m.c.0.3`	VALIDITY	Validity bit is used to indicate that the counter current value (frequency) and compare status registers contain valid data. If the bit is set to 1, the data is valid. If the bit is set to 0, the data is not valid.
`%IWr.m.c.0.4`	HIGH_LIMIT	The bit is set to 1 when the input frequency signal is out of range.

Type of the IODDT

In this mode, the type of the IODDT must be T_UNSIGNED_CPT_BMX.

Operating Limits

The maximum frequency that the module can measure on the IN_A input is 60 kHz. Beyond 60 kHz, the counting register value may decrease until it reaches 0. Beyond 60 kHz and up to the real cut-off frequency of 100 kHz, the module may indicate that it has exceeded the frequency limit.

When there is a variation in frequency, the value restoration time is 1 s with a value precision of 1 Hz. When there is a very significant variation in frequency, an accelerator enables you to restore the frequency value with a precision of 10 Hz in 0.1 s.

The maximum duty cycle at 60 KHz is 60%.

NOTE: You have to check the `validity` bit (`%IWr.m.c.0.3`) before taking into account the numerical values such as the counter and the capture registers. Only the `validity` bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.

BMX EHC 0200 Module Operation in Event Counting Mode

At a Glance

Using the event counting mode allows you to determine the number of events received in a scattered manner.

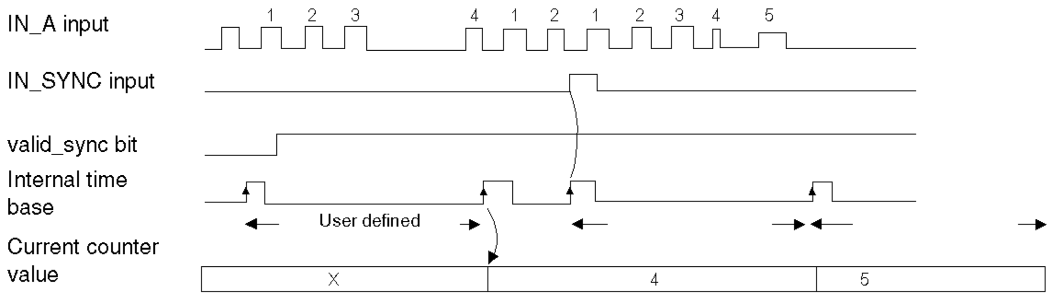
Basic Principle

In this mode, the counter assesses the number of pulses applied at the IN_A input, at time intervals defined by the user. The counting register is updated at the end of each interval with the number of events received.

It is possible to use the IN_SYNC input over a time interval, provided that the validation bit is set to 1. This restarts the event counting for a new predefined time interval. Depending on the selection made by the user, the time interval starts at the rising edge or at the falling edge on the IN_SYNC input.

Operation

The trend diagram below illustrates the counting process in event counting mode:



Counter Status Bits in Event Counting Mode

The table below shows the composition of the counter's `%IWr.m.c.0` status word in event counting mode:

Bit	Label	Description
`%IWr.m.c.0.2`	SYNC_REF_FLAG	The bit is set to 1 when the internal time base has been synchronized. The bit is set to 0 when the `sync_reset` command is received (rising edge of the `%Qr.m.c.8` bit).
`%IWr.m.c.0.3`	VALIDITY	Validity bit is used to indicate that the counter current value (events number) and compare status registers contain valid data. If the bit is set to 1, the data is valid. If the bit is set to 0, the data is not valid.
`%IWr.m.c.0.4`	HIGH_LIMIT	The bit is set to 1 when the number of received events exceeds the counter size. The bit is reset to 0 at the next period if the limit is not reached.
`%IWr.m.c.0.5`	LOW_LIMIT	The bit is set to 1 when more than one synchronization is received within 5 ms period. The bit is reset to 0 at the next period if the limit is not reached.

Type of the IODDT

In this mode, the type of the IODDT must be T_UNSIGNED_CPT_BMX.

Operating Limits

The module counts the pulses applied at the IN_A input every time the pulse duration is greater than 5 µs (without debounce filter).

The synchronization of the counter must not be done more than one time per 5 ms.

NOTE: You have to check the `validity` bit (`%IWr.m.c.0.3`) before taking into account the numerical values such as the counter and the capture registers. Only the `validity` bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.

BMX EHC 0200 Module Operation in Period Measuring Mode

At a Glance

Using the period measuring mode allows to:

- determine the duration of an event.
- determine the time between two events.
- set and measure the execution time for a process.

Basic Principle

This counting mode consists of two sub-modes:

- Rising edge to falling edge mode (edge to opposite): allows you to measure the duration of an event.
- Rising edge to rising edge mode (edge to edge): allows you to measure the length of time between two events.

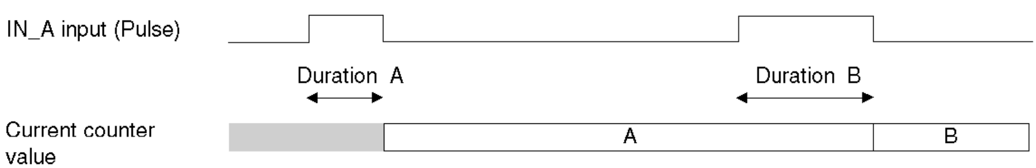
The user may also use the IN_SYNC input to enable or stop a measurement. It is also possible to specify a time out value in the configuration screen. This function allows to stop a measurement that exceeds this time out. In this case, the counting register is not valid until the next complete measurement.

The units used to measure the length of time of an event or between two events are defined by the user (1 µs, 100 µs or 1 ms).

Edge to Opposite Mode

In this sub-mode, the measurement is taken between the rising edge and the falling edge of the IN_A input. The counting register is updated as soon as the falling edge is detected.

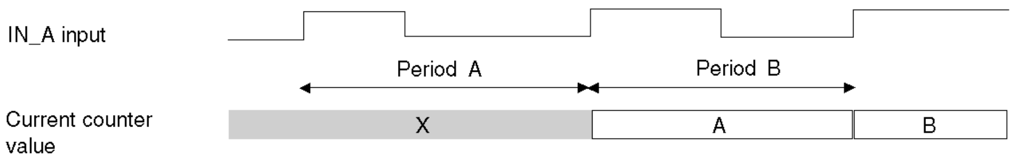
The trend diagram below shows the operating mode of the edge to opposite sub-mode:



Edge to Edge Mode

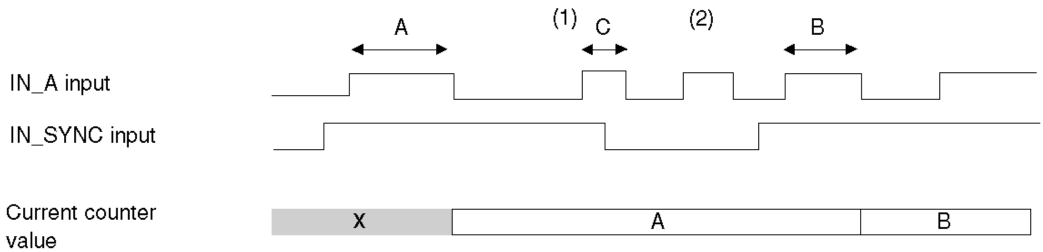
In this sub-mode, the measurement is taken between two rising edges of the IN_A input. The counting register is updated as soon as the second rising edge is detected.

The trend diagram below shows the operating mode of the edge to edge sub-mode:



Using the Synchronization Function

The trend diagram below illustrates the period measurement counting process in edge to opposite mode when using the synchronization function:



(1) The falling edge of the IN_SYNC input stops measurement C.
(2) This pulse is not measured because the IN_SYNC input is not at the high level.

NOTE: The `valid_sync` bit (`%QWr.m.c.0.0`) must be set to 1 to enable the IN_SYNC input. If the IN_SYNC input is not wired, the application must force the setting of the `force_sync` bit (`%Qr.m.c.4`) to 1 to authorize the measurements.

Counter Status Bits in Period Measuring Mode

The table below shows the composition of the counter's `%IWr.m.c.0` status word in period measuring mode:

Bit	Label	Description
`%IWr.m.c.0.3`	VALIDITY	Validity bit is used to indicate that the counter current value (period value) and compare status registers contain valid data. If the bit is set to 1, the data is valid. If the bit is set to 0, the data is not valid.
`%IWr.m.c.0.4`	HIGH_LIMIT	The bit is set to 1 when the measured period exceeds the user-defined timeout. The bit is reset to 0 at the next period if the timeout is not reached.
`%IWr.m.c.0.5`	LOW_LIMIT	The bit is set to 1 when more than one measure occurs within 5 ms period. The bit is reset to 0 at the next period if the limit is not reached.

Type of the IODDT

In this mode, the type of the IODDT must be T_UNSIGNED_CPT_BMX.

Operating Limits

The module can perform a maximum of 1 measurement every 5 ms.

The shortest pulse that can be measured is 100 µs, even if the unit defined by the user is 1 µs.

The maximum duration that can be measured is 1,073,741,823 units of time (unit defined by the user).

NOTE: You have to check the `validity` bit (`%IWr.m.c.0.3`) before taking into account the numerical values such as the counter and the capture registers. Only the `validity` bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.

BMX EHC 0200 Module Operation in Ratio Mode

At a Glance

The ratio mode only uses the IN_A and IN_B inputs. This counting mode consists of two sub-modes:

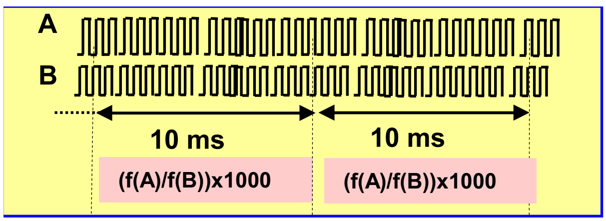
- Ratio 1: is used to divide two frequencies (Frequency IN_A / Frequency IN_B) and is useful, for example, in applications such as flowmeters and mixers.
- Ratio 2: is used to subtract two frequencies (Frequency IN_A - Frequency IN_B) and is used in the same applications but which require more precise adjustment (closer frequencies).

NOTE: A positive value indicates that the frequency measured on the IN_A input is greater than the frequency measured on the IN_B input.

A negative value indicates that the frequency measured on the IN_A input is less than the frequency measured on the IN_B input.

Ratio 1 Mode

The figure below shows BMX EHC 0200 module operation in Ratio 1 mode.



In this mode, the counter evaluates the ratio between the number of rising edges of the IN_A input and the number of rising edges of the IN_B input over a period of 1 s. The register value is updated every 10 ms.

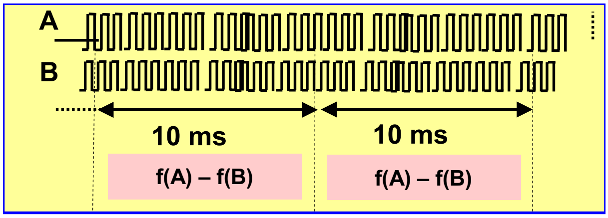
An absolute limit value is declared on the configuration screen. If this limit value is exceeded, the `counter_value` register (`%IDr.m.c.12`) is disabled by setting the `validity` bit (`%IWr.m.c.0.3`) to 0.

If no frequency is applied to the IN_A or IN_B inputs, the `counter_value` register (`%IDr.m.c.12`) is disabled by setting the `validity` bit (`%IWr.m.c.0.3`) to 0.

NOTE: The ratio 1 mode presents the results in thousandths in order to have greater level of precision (where 2,000 is displayed, this corresponds to a value of 2).

Ratio 2 Mode

The figure below shows BMX EHC 0200 module operation in Ratio 2 mode.



In this mode, the counter evaluates the difference between the number of rising edges of the IN_A input and the number of rising edges of the IN_B input over a period of 1 s. The `counter_value` register (`%IDr.m.c.12`) is updated at the end of each 10 ms interval.

An absolute limit value is declared on the configuration screen. If this limit value is exceeded, the `counter_value` register (`%IDr.m.c.12`) is disabled and the `validity` bit (`%IWr.m.c.0.3`) to 0.

Counter Status Bits in Ratio Mode

The table below shows bits that are used by the status word `%IWr.m.c.0` when the counter is configured in ratio mode:

Bit	Label	Description
`%IWr.m.c.0.3`	VALIDITY	Validity bit is used to indicate that the counter current value (ratio value) and compare status registers contain valid data. If the bit is set to 1, the data is valid. If the bit is set to 0, the data is not valid.
`%IWr.m.c.0.4`	HIGH_LIMIT	The bit signals a error when the ratio exceeds the absolute limit. The bit is set to 1 when frequency to IN_A becomes too fast. The bit is reset to 0 when the frequency to IN_A remains correct.
`%IWr.m.c.0.5`	LOW_LIMIT	The bit signals a error when the ratio exceeds the absolute limit. The bit is set to 1 when frequency to IN_B becomes too fast. The bit is reset to 0 when the frequency to IN_B remains correct.

Type of the IODDT

In this mode, the type of the IODDT must be T_SIGNED_CPT_BMX.

Operating Limits

The maximum frequency that the module can measure on the IN_A and IN_B inputs is 60 kHz.

The measured values are between -60,000,000,000 and +60,000,000,000.

NOTE: You have to check the `validity` bit (`%IWr.m.c.0.3`) before taking into account the numerical values such as the counter and the capture registers. Only the `validity` bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.

BMX EHC 0200 Module Operation in One Shot Counter Mode

At a Glance

Using the one shot counter mode allows you to quantify a group of parts.

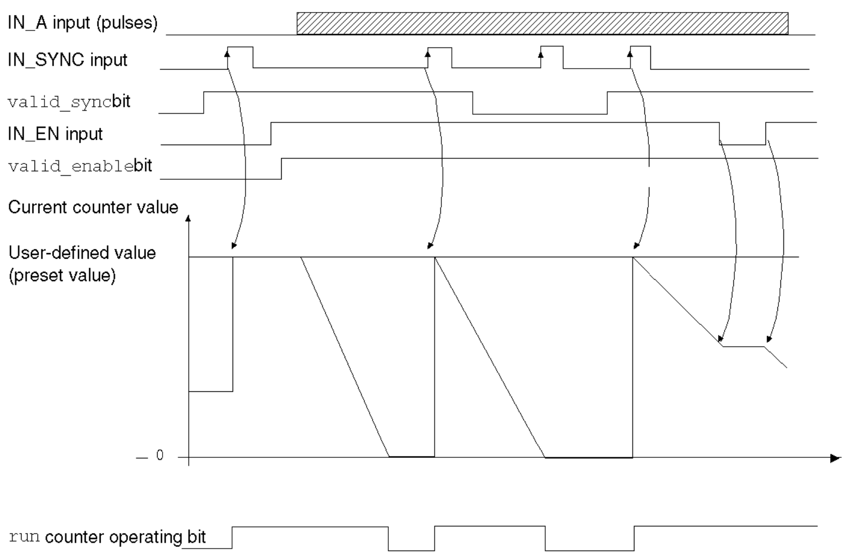
Basic Principle

In this mode, activating the synchronization function starts the counter which, starting from a value defined by the user in the adjust screen (`preset value`), decreases with every pulse applied to the IN_A input until it reaches the value 0. Downcounting is made possible when the enable function is activated. The counting register is thus updated every 1 ms.

One basic use of this mode is, using an output, to indicate the end of a group of operations (when the counter reaches 0).

Operation

The trend diagram below illustrates the counting process in one shot counter mode:



In the trend diagram above, we can see that the counter is set to the preset value at the IN_SYNC input’s rising edge. Then, the counter decrements the counting register with every pulse applied to the IN_A input. When the register is set to 0, the counter awaits a new signal from the IN_SYNC input. The IN_A input pulses have no effect on the register value as long as the counter is set to 0.

The enable function must be activated during the counting by:

- setting to 1 the force_enable bit
- setting to 1 the valid_enable bit and when the IN_EN input is at the high level

When the enable function is deactivated, the last value reported in the counting register is maintained and the counter ignores the pulses applied to the IN_A input. However, it does not ignore the IN_SYNC input status.

Each time the counter starts a downcounting operation, the run bit switches to the high level. It switches to the low level when the register value reaches 0.

NOTE: The pulses applied to IN_SYNC and IN_EN inputs are only taken into account when the inputs are enabled *(see page 64)*.

The value defined by the user (preset value) is contained in the word %MDr.m.c.6. The user may change this value by specifying the value of this word by configuring the parameter in the adjust screen or by using the WRITE_PARAM(IODDT_VAR1) Function. IODDT_VAR1 is of the type T_UNSIGNED_CPT_BMX. This value change is only taken into account by the module after one of the following conditions has been established:

- At the next synchronization if the counter is stopped (run bit set to 0)
- At the second synchronization if the counter is activated (run bit set to 1).

Counter Status Bits in One shot Counter Mode

The table below shows bits that are used by the status word %IWr.m.c.0 when the counter is configured in one shot counter mode:

Bit	Label	Description
%IWr.m.c.0.0	RUN	The bit is set to 1 when the counter is running. The bit is set to 0 when the counter is stopped.
%IWr.m.c.0.2	SYNC_REF_FLAG	The bit is set to 1 when the counter has been set to the preset value and (re)started. The bit is reset to 0 when the sync_reset command is received (rising edge of the %Qr.m.c.8 bit).
%IWr.m.c.0.3	VALIDITY	Validity bit is used to indicate that the counter current value and compare status registers contain valid data. If the bit is set to 1, the data is valid. If the bit is set to 0, the data is not valid.

Type of the IODDT

In this mode, the type of the IODDT must be T_UNSIGNED_CPT_BMX.

Operating Limits

The maximum frequency that can be applied to the IN_SYNC input is 1 pulse every 5 ms.

The maximum value defined by the user (preset value) is 4,294,967,295.

NOTE: You have to check the `validity` bit (`%IWr.m.c.0.3`) before taking into account the numerical values such as the counter and the capture registers. Only the `validity` bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.

BMX EHC 0200 Module Operation in Modulo Loop Counter Mode

At a Glance

The use of the modulo loop counter mode is recommended for packaging and labeling applications for which actions are repeated for series of moving objects.

Basic Principle

In the upcounting direction, the counter increases until it reaches the modulo value -1, the modulo value being defined by the user. At the following pulse in the counting direction, the counter is reset to 0 and the counting resumes.

In the downcounting direction, the counter decreases until it reaches 0. At the next pulse in the counting direction, the counter is reset to the the modulo value -1, the modulo value being defined by the user. The downcounting may then be resumed.

The enable function must be activated during the counting by:

- Setting to 1 the `force_enable` bit (`%Qr.m.c.6`)
- Setting to 1 the `valid_enable` bit (`%QWr.m.c.0.2`) when the IN_EN input is at the high level

When the enable function is deactivated, the last value reported in the counting register is maintained and the counter ignores the pulses applied to the IN_A input. However, it does not ignore the preset condition.

In the modulo loop counter mode, the counter must be synchronized at least one time to operate. The current counter value is cleared each time the synchronization occurs.

The current counter value can be recorded into the capture0 register *(see page 66)* when the condition of synchronization occurs *(see page 60)*.

The modulo value defined by the user is contained in the `modulo_value` word `%MDr.m.c.4`. The user may change this value by specifying the value of this word:

- In the ajust screen
- In the application, using the `WRITE_PARAM(IODDT_VAR1)` Function. `IODDT_VAR1` is of the type `T_UNSIGNED_CPT_BMX`.

The new modulo value is acknowledged if one of the two following conditions is met:

- The synchronization is activated
- The counter rollovers the value 0 in the downcounting direction or the modulo value -1 (this value is the modulo value recorded before editing the new modulo value) in the upcounting direction.

Counting Interface

In this mode, the user may select one of the following counting configurations:

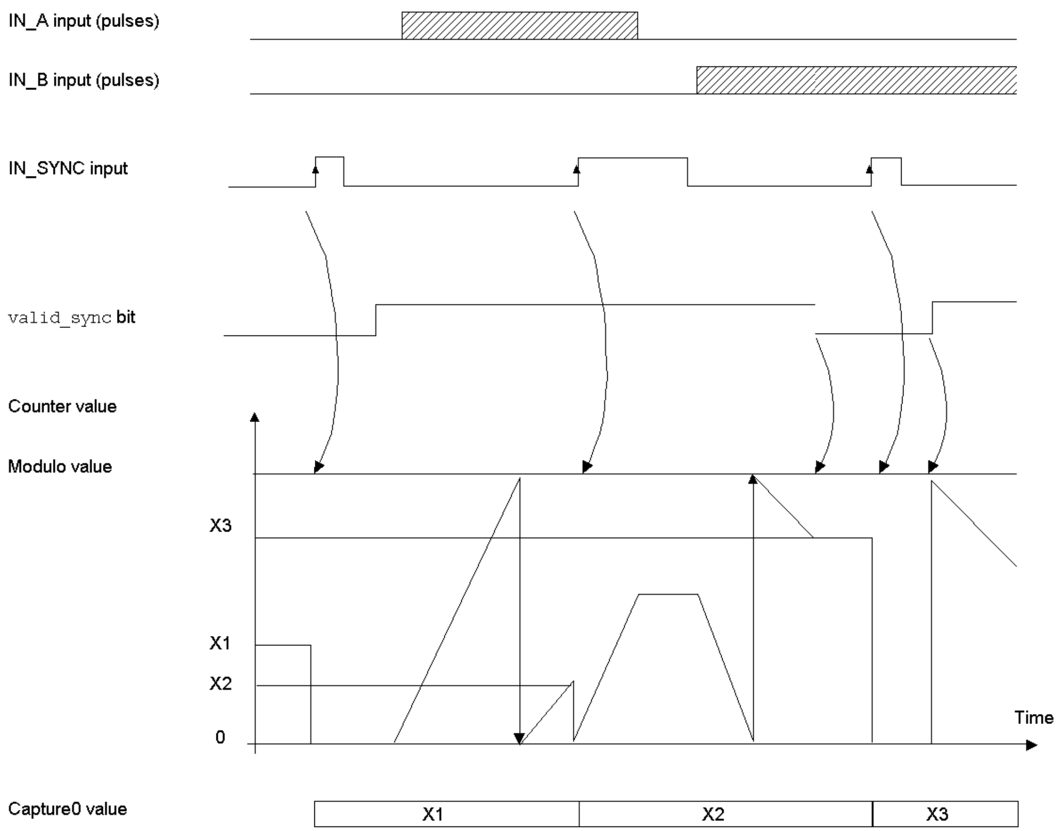
- A = Up, B = Down (default configuration)
- A = Impulse, B = Direction
- Normal Quadrature X1
- Normal Quadrature X2
- Normal Quadrature X4
- Reverse Quadrature X1
- Reverse Quadrature X2
- Reverse Quadrature X4.

The following table shows the upcounting and downcounting principle according to the selected configuration:

Selected configuration	Upcounting condition	Downcounting condition
A = Up, B = Down	Rising edge at the IN_A input.	Rising edge at the IN_B input.
A = Impulse, B = Direction	Rising edge at the IN_A input and low state at the IN_B input.	Rising edge at the IN_A input and high state at the IN_B input.
Normal Quadrature X1	Rising edge at the IN_A input and low state at the IN_B input.	Falling edge at the IN_A input and low state at the IN_B input
Normal Quadrature X2	Rising edge at the IN_A input and low state at the IN_B input. Falling edge at the IN_A input and high state at the IN_B input.	Falling edge at the IN_A input and low state at the IN_B input. Rising edge at the IN_A input and high level at the IN_B input.
Normal Quadrature X4	Rising edge at the IN_A input and low state at the IN_B input. High state at the IN_A input and rising edge at the IN_B input. Falling edge at the IN_A input and high state at the IN_B input. Low state at the IN_A input and falling edge at the IN_B input.	Falling edge at the IN_A input and low state at the IN_B input. Low state at the IN_A input and rising edge at the IN_B input. Rising edge at the IN_A input and high level at the IN_B input. High state at the IN_A input and falling edge at the IN_B input.
Reserve Quadrature X1	Falling edge at the IN_A input and low state at the IN_B input.	Rising edge at the IN_A input and low state at the IN_B input.
Reserve Quadrature X2	Falling edge at the IN_A input and low state at the IN_B input. Rising edge at the IN_A input and high level at the IN_B input.	Rising edge at the IN_A input and low state at the IN_B input. Falling edge at the IN_A input and high state at the IN_B input.
Reserve Quadrature X4	Falling edge at the IN_A input and low state at the IN_B input. Low state at the IN_A input and rising edge at the IN_B input. Rising edge at the IN_A input and high level at the IN_B input. High state at the IN_A input and falling edge at the IN_B input.	Rising edge at the IN_A input and low state at the IN_B input. High state at the IN_A input and rising edge at the IN_B input. Falling edge at the IN_A input and high state at the IN_B input. Low state at the IN_A input and falling edge at the IN_B input.

Operation

The trend dithe modulo counting process in the configuration by default (IN_A = counting, In_B = downcounting):



Counter Status Bits in Modulo Loop Counter Mode

The table below shows the composition of the counter's `%IWr.m.c.0` status word in modulo loop counter mode:

Bit	Label	Description
`%IWr.m.c.0.1`	MODULO_FLAG	The bit is set to 1 when the counter rollovers the modulo and is . The bit is reset to 0 when the command `MODULO_RESET` (`%Qr.m.c.9`) is received (rising edge of the `MODULO_RESET` bit).
`%IWr.m.c.0.2`	SYNC_REF_FLAG	The bit is set to 1 when the counter have been set to 0 and (re)started. The bit is reset to 0 when the command `SYNC_RESET` (`%Qr.m.c.8`) is received (rising edge of the `SYNC_RESET` bit).
`%IWr.m.c.0.3`	VALIDITY	Validity bit is used to indicate that the counter current value and compare status registers contain valid data. If the bit is set to 1, the data is valid. If the bit is set to 0, the data is not valid.

Type of the IODDT

In this mode, the type of the IODDT must be T_UNSIGNED_CPT_BMX.

Operating Limits

The maximum frequency that can be applied to the IN_SYNC input is 1 pulse every 5 ms.

The maximum frequency for the modulo event is once every 5 ms.

The maximum value for the defined modulo value and the counter is 4,294,967,295.

NOTE: You have to check the `validity` bit (`%IWr.m.c.0.3`) before taking into account the numerical values such as the counter and the capture registers. Only the `validity` bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.

BMX EHC 0200 Module Operation in Free Large Counter Mode

At a Glance

The use of the free large counter mode is especially recommended for axis monitoring or labeling where the incoming position of each part has to be learned.

Basic Principle

The upcounting (or downcounting) starts as soon as the homing function is completed.

The enable function must be activated during the counting by:

- Setting to 1 the force_enable bit (`%Qr.m.c.6`)
- Setting to 1 the valid_enable bit (`%QWr.m.c.0.2`) when the IN_EN input is at the high level.

When the enable function is deactivated, the last value reported in the counting register is maintained and the counter ignores the pulses applied to the IN_A input. However, it does not ignore the preset condition.

In the free large counter mode, the counter must be preset at least one time to operate. The current counter value is load with the `preset_value` each time the preset condition occurs.

The current counter can be recorded into the capture0 register when the preset condition occurs or using the IN_CAP input.

The current counter value can be stored into the capture1 register using the IN_CAP input.

For further information, you may see the synchronization function *(see page 60)* and the capture function *(see page 66)*.

In the free large counter mode, the counting register is updated at 1 ms intervals.

Counting Configurations

In this mode, the user may select one of the following counting configurations:

- A = Up, B = Down (default configuration)
- A = Impulse, B = Direction
- Normal Quadrature X1
- Normal Quadrature X2
- Normal Quadrature X4
- Reverse Quadrature X1
- Reverse Quadrature X2
- Reverse Quadrature X4

The following table shows the upcounting and downcounting principle according to the selected configuration:

Selected configuration	Upcounting condition	Downcounting condition
A = Up, B = Down	Rising edge at the IN_A input.	Rising edge at the IN_B input.
A = Impulse, B = Direction	Rising edge at the IN_A input and low state at the IN_B input.	Rising edge at the IN_A input and high state at the IN_B input.
Normal Quadrature X1	Rising edge at the IN_A input and low state at the IN_B input.	Falling edge at the IN_A input and low state at the IN_B input
Normal Quadrature X2	Rising edge at the IN_A input and low state at the IN_B input. Falling edge at the IN_A input and high state at the IN_B input.	Falling edge at the IN_A input and low state at the IN_B input. Rising edge at the IN_A input and high level at the IN_B input.
Normal Quadrature X4	Rising edge at the IN_A input and low state at the IN_B input. High state at the IN_A input and rising edge at the IN_B input. Falling edge at the IN_A input and high state at the IN_B input. Low state at the IN_A input and falling edge at the IN_B input.	Falling edge at the IN_A input and low state at the IN_B input. Low state at the IN_A input and rising edge at the IN_B input. Rising edge at the IN_A input and high level at the IN_B input. High state at the IN_A input and falling edge at the IN_B input.
Reserve Quadrature X1	Falling edge at the IN_A input and low state at the IN_B input.	Rising edge at the IN_A input and low state at the IN_B input.
Reserve Quadrature X2	Falling edge at the IN_A input and low state at the IN_B input. Rising edge at the IN_A input and high level at the IN_B input.	Rising edge at the IN_A input and low state at the IN_B input. Falling edge at the IN_A input and high state at the IN_B input.
Reserve Quadrature X4	Falling edge at the IN_A input and low state at the IN_B input. Low state at the IN_A input and rising edge at the IN_B input. Rising edge at the IN_A input and high level at the IN_B input. High state at the IN_A input and falling edge at the IN_B input.	Rising edge at the IN_A input and low state at the IN_B input. High state at the IN_A input and rising edge at the IN_B input. Falling edge at the IN_A input and high state at the IN_B input. Low state at the IN_A input and falling edge at the IN_B input.

Homing Function

This function allows to record the `current_counter_value` register in the `capt_0_val` register and/or to set the `current_counter_value` register to the user-predefined parameter `preset_value`.

The value defined by the user as `preset_value` is contained in the `%MDr.m.c.4` word.

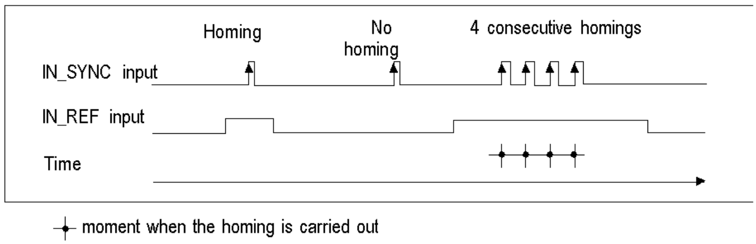
The user may change this value by specifying the value of this word:

- In the ajust screen
- In the application, by using the `WRITE_PARAM(IODDT_VAR1)` Function. `IODDT_VAR1` is of the type `T_SIGNED_CPT_BMX`.

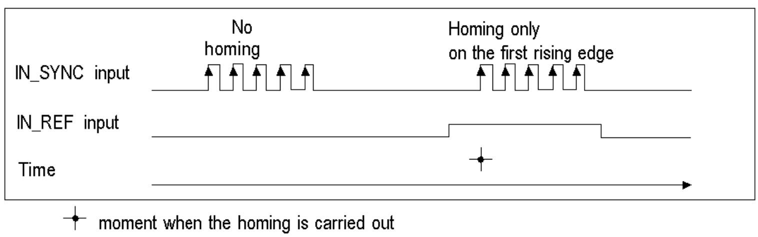
For further information, you may see the homing function *(see page 62)* and the capture function *(see page 66)*.

The module configuration enables you to select the following homing conditions:

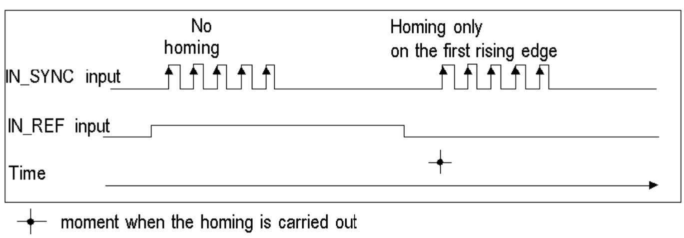
- Rising edge of the IN_SYNC input (default)
- Rising edge of the IN_REF input
- Rising edge of the IN_SYNC input at the IN_REF input's high level:



- First rising edge of the IN_SYNC input and high level at the IN_REF input

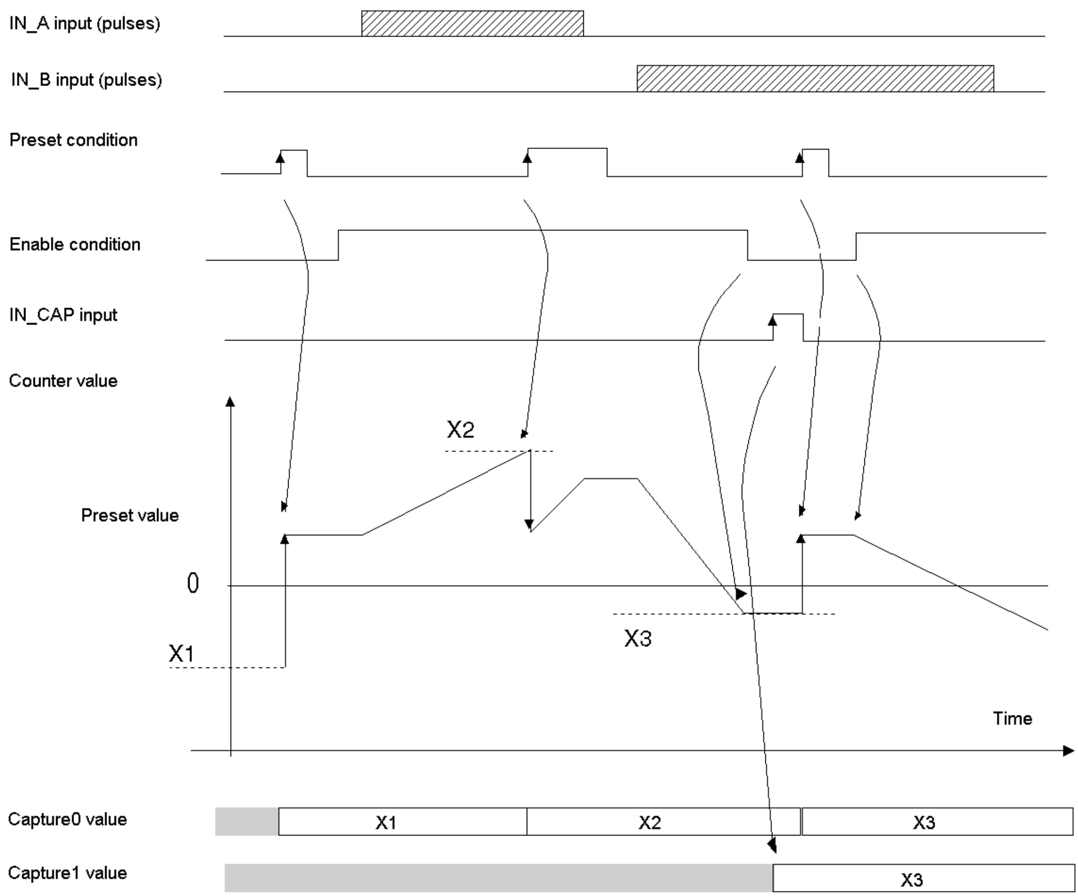


- First rising edge of the IN_SYNC input and low level at the IN_REF input



Operation

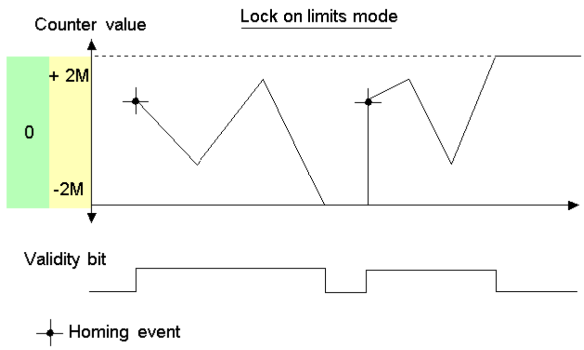
The trend diagram below illustrates the counting process for a free large counter in the configuration by default:



Behavior at the Counting Limits

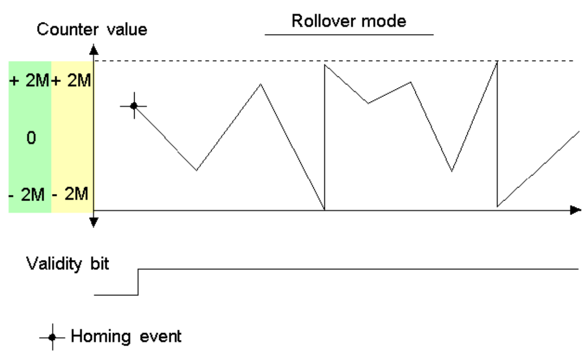
When the upper or lower limit is exceeded, the counter behaves differently according to its configuration.

In the lock on limits default configuration, the counting register maintains the limit value once it has been reached and the counting validity bit changes to 0 until the next preset condition occurs:



NOTE: Overflow or underflow is indicated by two bits `LOW_LIMIT` and `HIGH_LIMIT` until the application reloads the counting value predefined by the user (`force_ref` bit set to 1 or preset condition true). The upcounting or downcounting may therefore resume.

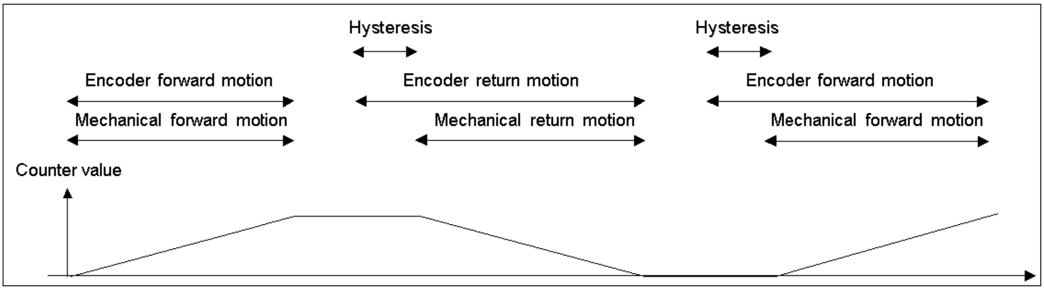
In the rollover configuration, the counting register switches to the opposite limit value when one of the two limits is exceeded:



Slack Delete

In the free large counter mode, the counter may apply a hysteresis if the rotation is inverted. The `hysteresis` parameter configured with the adjust screen defines the number of points that are not acknowledged by the counter during the rotation inversion. This aims to take into account the slack between the encoder/motor axis and the mechanical axis (e.g. an encoder measuring the position of a mat).

This behavior is described in the following figure:



The value defined by the user as the `Hysteresis (slack)` value is contained in the `%MWr.m.c.9` word. The user may change this value by specifying the value of this word (this value is from 0 to 255):

- In the adjust screen
- In the application by using the `WRITE_PARAM(IODDT_VAR1)` Function. `IODDT_VAR1` is of the type `T_SIGNED_CPT_BMX`.

Counter Status Bits in Free Large Counter Mode

The table below shows the composition of the counter's `%IWr.m.c.0` status word in free large counter mode:

Bit	Label	Description
`%IWr.m.c.0.1`	MODULO_FLAG	The bit status changes in the rollover mode. The bit is set to 1 when the counter rollovers its limits (-2,147,483,648 or +2,147,483,647). The bit is reset to 0 when the command `MODULO_RESET` (`%Qr.m.c.9`) is received (rising edge of the `MODULO_RESET` bit).
`%IWr.m.c.0.2`	SYNC_REF_FLAG	The bit is set to 1 when the counter have been set to the preset value and (re)started. The bit is reset to 0 when the command `SYNC_RESET` (`%Qr.m.c.8`) is received (rising edge of the `SYNC_RESET` bit).

Bit	Label	Description
%IWr.m.c.0.3	VALIDITY	Validity bit is used to indicate that the counter current value and compare status registers contain valid data. If the bit is set to 1, the data is valid. If the bit is set to 0, the data is not valid.
%IWr.m.c.0.4	HIGH_LIMIT	The bit status changes in the lock on limits mode. The bit is set to 1 when the counter reaches +2,147,483,647. The bit is reset to 0 when the counter presets or resets.
%IWr.m.c.0.5	LOW_LIMIT	The bit status changes in the lock on limits mode. The bit is set to 1 when the counter reaches -2,147,483,648. The bit is reset to 0 when the counter presets or resets.

Type of the IODDT

In this mode, the type of the IODDT must be T_SIGNED_CPT_BMX.

Operating Limits

The shortest pulse applied to the IN_SYNC input is 100 µs.

The maximum homing event frequency is once every 5 ms.

The counter value is between -2,147,483,648 and +2,147,483,647.

NOTE: You have to check the `validity` bit (`%IWr.m.c.0.3`) before taking into account the numerical values such as the counter and the capture registers. Only the `validity` bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.

BMX EHC 0200 Module Operation in Pulse Width Modulation Mode

At a Glance

In this operating mode, the module uses an internal clock generator to supply a periodic signal at the module's Q0 output. Only the Q0 output is concerned by this mode as the Q1 output is independent of this mode.

Basic Principle

The `output_block_0_enable` command bit (`%Qr.m.c.2`) must be set to 1 in order to enable a modulation at the Q0 output.

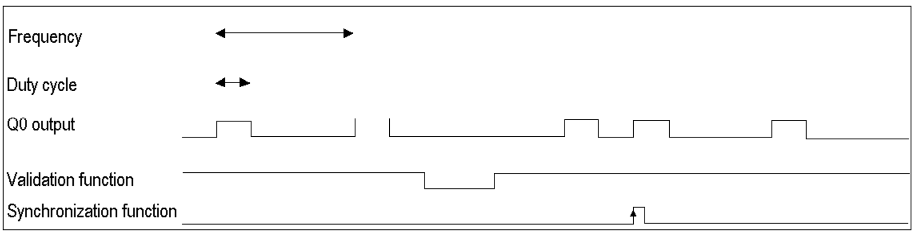
The active validation function enables the operation of the internal clock generator that produces the output signal to be validated.

The active synchronization function enables the output signal to be synchronized by resetting to 0 the internal clock generator.

The wave form of the output signal depends on:

- The `pwm_frequency` value (`%QDr.m.c.6`): it defines the frequency from 0.1 Hz (value is equal to 1) to 4 KHz (value is equal to 40,000), in increments of 0.1 Hz
- The `pwm_duty` value (`%QWr.m.c.8`): it defines the duty cycle from 5 % (value is equal to 1) to 95 % (value is equal to 19) in increments of 5 %.

The following figure shows the operation of the module in the pulse width modulation mode:



Counter Status Bits in Pulse Width Modulation Mode

The table below shows the composition of the counter's `%IWr.m.c.0` status word in pulse width modulation mode:

Bit	Label	Description
`%IWr.m.c.0.3`	VALIDITY	Validity bit is used to indicate that the output data (frequency and duty cycle) unter current value and compare status registers contain valid data. If the bit is set to 1, the data is valid. If the bit is set to 0, the data is not valid.

Bit	Label	Description
%IWr.m.c.0.4	HIGH_LIMIT	The output frequency or the duty cycle is out of range (high limit).
%IWr.m.c.0.5	LOW_LIMIT	The output frequency or the duty cycle is out of range (low limit).

Type of the IODDT

In this mode, the type of the IODDT must be T_UNSIGNED_CPT_BMX.

Operating Limits

The maximum output frequency is 4 kHz.

The maximum frequency that can be applied to the IN_SYNC input is 1 pulse every 5 ms.

The Q0 driver is "source type", therefore a load resistance is required to switch the output signal Q0 to 0 V using the correct frequency. We recommend a load resistance of 250 Ω

The allowed duty cycle varies according to the frequency of the Q0 output.

The table below shows duty cycle values according to the selected frequency. These values must be observed for normal operation:

Frequency	Duty cycle
0.1... 250 Hz	95% - 5%
251... 500 Hz	90% - 10%
501... 1 000 Hz	80% - 20%
1001... 1 500 Hz	70% - 30%
1501... 2 000 Hz	60% - 40%
2 001... 2 500 Hz	50%
2 5001... 4 000 Hz	50% (See following note)

NOTE: If the frequency and the duty cycle do not respect this table, the output and the `validity` bit (`%IWr.m.c.0.3`) remains in the low state.

NOTE: You have to check the `validity` bit (`%IWr.m.c.0.3`) before taking into account the numerical values such as the counter and the capture registers. Only the `validity` bit at the high level (set to 1) guarantees that the mode will operate correctly within the limits.

NOTE: From 2501 Hz to 4000 Hz, the 50% ratio is not guaranteed on output.

Counting Module BMX EHC 0200 Software Implementation

IV

Subject of this Part

This part describes the software implementation and functions of the BMX EHC 0200 counting module.

NOTE: This part concerns also the Modicon M340H.

What Is in This Part?

This part contains the following chapters:

Software Implementation Methodology for BMX EHC xxxx Counting Modules

7

Installation Methodology

At a Glance

The software installation of the BMX EHC **** counting modules is carried out from the various Unity Pro editors:

- in offline mode,
- in online mode.

The following order of installation phases is recommended but it is possible to change the order of certain phases (for example, starting with the configuration phase).

Installation Phases

The following table shows the different installation phases:

Phase	Description	Mode
Declaration of variables	Declaration of IODDT-type variables for the application-specific modules and variables of the project.	Offline(1)
Programming	Project programming.	Offline(1)
Configuration	Declaration of modules.	Offline
	Module channel configuration	
	Entering the configuration parameters **Note:** All the parameters are configurable online except the `event` parameter.	Offline(1)
Association	Association of IODDTs with the channels configured (variable editor)	Offline(1)
Build	Project generation (analysis and editing of links)	Offline
Transfer	Transfer project to PLC	Online

Phase	Description	Mode
Adjustment/Debugging	Debug project from debug screens, animation tables	Online
	Debugging the program and adjustment parameters	
Documentation	Building documentation file and printing miscellaneous information relating to the project	Online(1)
Operation/Diagnostic	Displaying miscellaneous information necessary for supervisory control of the project	Online
	Diagnostics of project and modules	
Key:		
(1)	These various phases can also be performed in online mode	

Accessing the Functional Screens of the BMX EHC xxxx Counting Modules

8

Subject of this Chapter

This chapter describes the various functional screens of the BMX EHC •••• counting modules to which the user has access.

What Is in This Chapter?

This chapter contains the following topics:

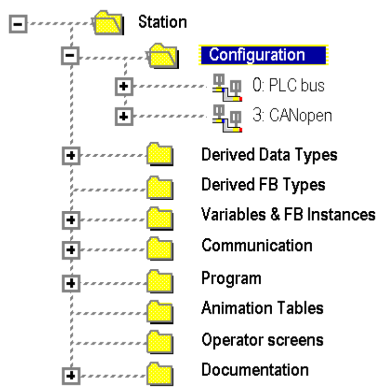
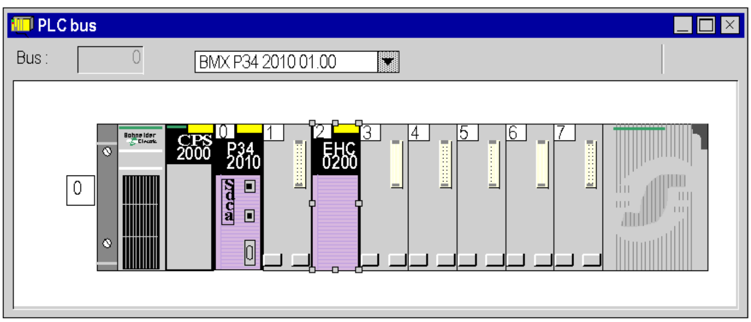
Accessing the Functional Screens of the BMX EHC 0200 Counting Modules

At a Glance

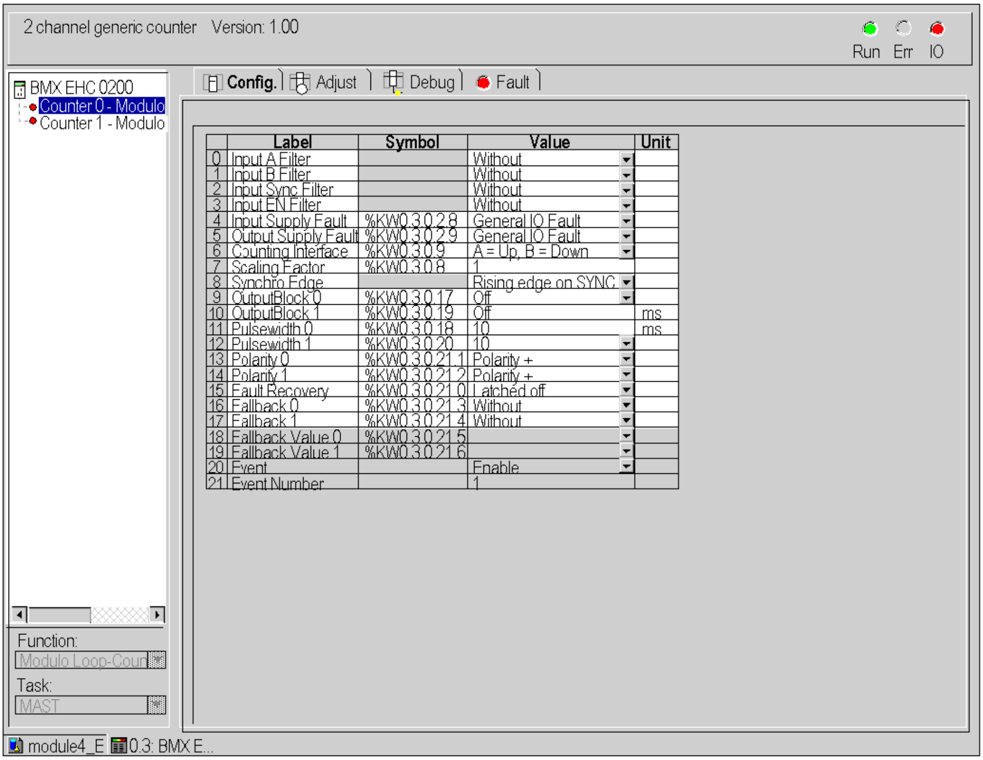
This section describes how to access the functional screens of the BMX EHC 0200 counting modules.

Procedure

To access the screens, execute the following actions:

Step	Action
1	Expand the Configuration directory in the project browser. **Result**: the following screen appears: 
2	Double-click on the PLC Bus directory. **Result**: the following screen appears: 

Step	Action
3	Double-click on the counting module. **Result**: the module screen appears:



Description of the Counting Module Screens

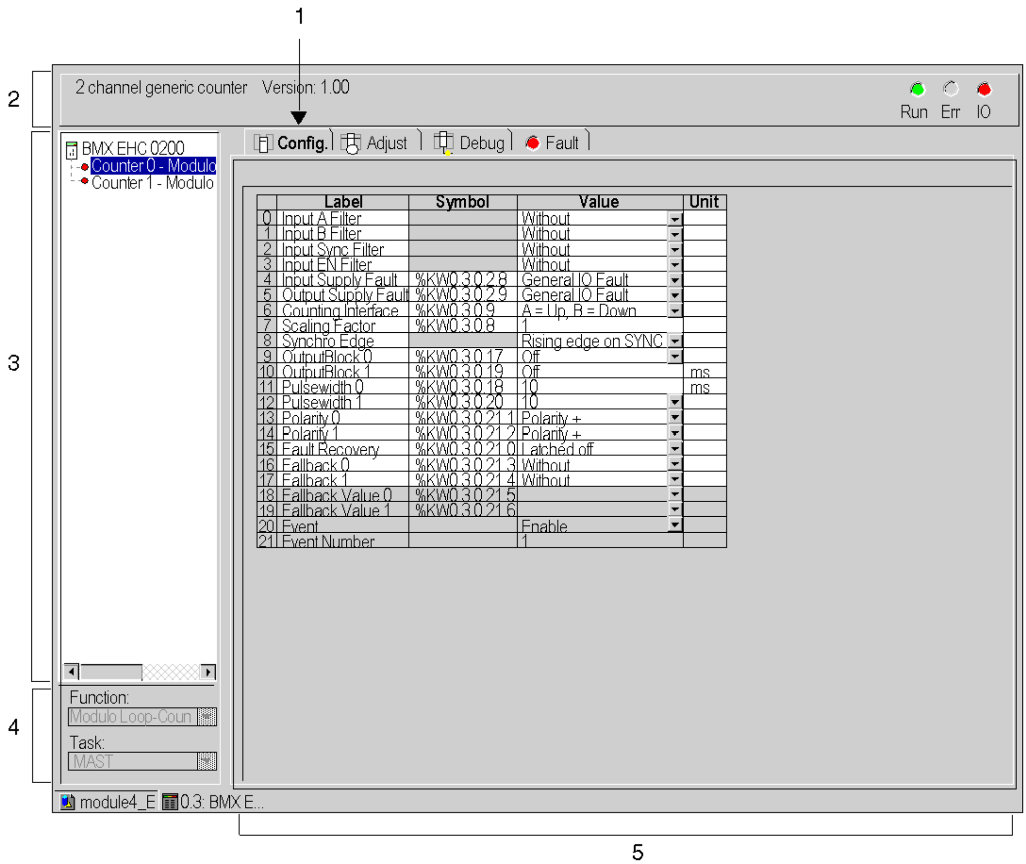
Introduction

The various available screens for the BMX EHC 0200 counting modules are:

- Configuration screen
- Adjust screen
- Debug screen (can only be accessed in online mode)
- Faults screen (can only be accessed in online mode)

Description of the Screens

The following diagram presents the counting modules configuration screen.



The following table presents the parts of the various screens.

Number	Element	Function
1	Tabs	The tab in the foreground indicates the mode in progress (**Configuration** in this example). Every mode can be selected using the respective tab. The available modes are: ● **Configuration** ● **Adjust** ● **Debug** (which can only be accessed in online mode) ● **Faults** (which can only be accessed in online mode)
2	**Module** area	Provides an abbreviation as a reminder of the module and module status in online mode (LEDs).

3	**Channel** area	Is used:
		● By clicking on the reference number, to display the tabs:
		  ● **Description** which gives the characteristics of the device.
		  ● **I/O Objects** which is used to presymbolize the input/output objects.
		  ● **Faults** which shows the device error (in online mode).
		● To select a channel.
		● To display the **Symbol**, name of the channel defined by the user (using the variable editor).

4	**General parameters** area	Allows you to select the counting function and the task associated with the channel: ● **Function:** counting function among those available for the modules involved. Depending on this choice, the headings of the configuration area may differ. By default, no function is configured. ● **Task:** defines the **MAST** or **FAST** task through which the channel's implicit exchange objects will be exchanged. These choices are only possible in offline mode.
5	**Parameters in progress** area	This area has various functionalities which depend upon the current mode: ● **Configuration**: is used to configure the channel parameters. ● **Adjust**: consists of various sections to be completed (parameter values), displayed according to the choice of counting function. ● **Debug**: displays the status of the inputs and outputs, as well as the various parameters of the current counting function. ● **Faults**: displays the error that have occurred on the counting channels.

Configuration of the BMX EHC 0200 Counting Modules

9

Subject of this Chapter

This chapter deals with the configuration of the BMX EHC 0200 counting modules. This configuration can be accessed from the Configuration tab on the functional screens of BMX EHC 0200 *(see page 108)* modules.

What Is in This Chapter?

This chapter contains the following sections:

9.1 Configuration Screen for BMX EHC xxxx Counting Modules

Subject of this Section

This section presents the configuration screen for BMX EHC •••• counting modules in a Modicon M340 local rack and in X80 drop.

What Is in This Section?

This section contains the following topics:

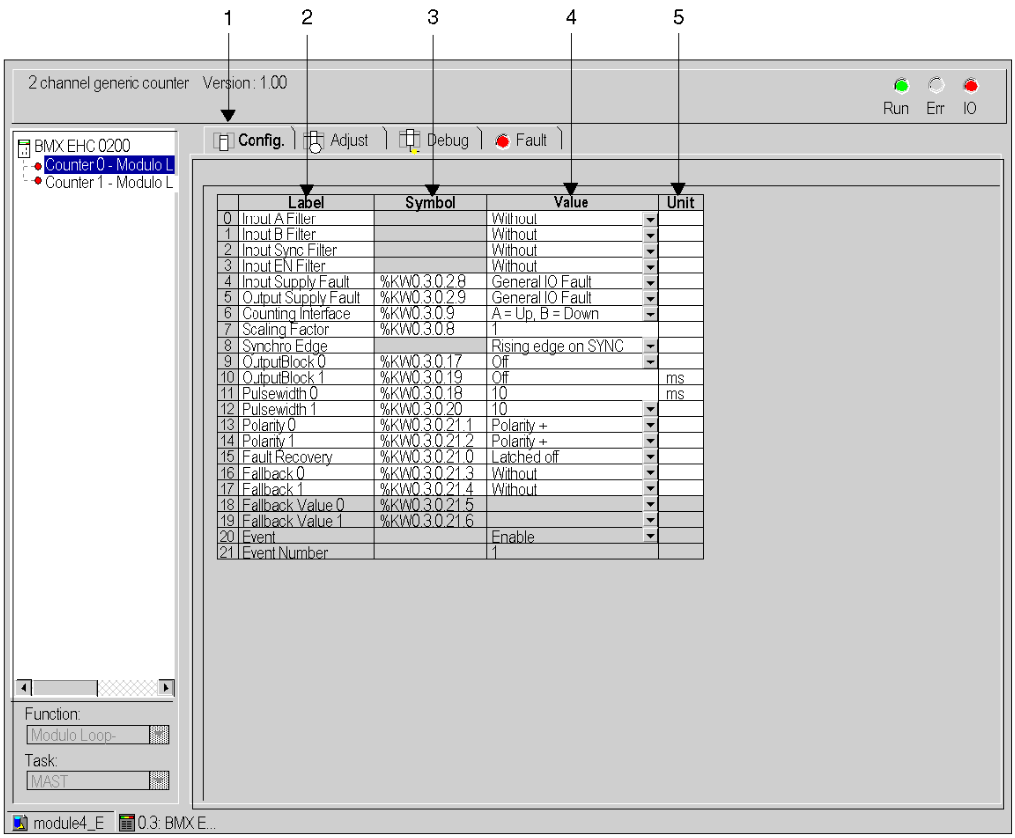
Configuration Screen for BMX EHC 0200 Counting Modules in a Modicon M340 Local Rack

At a Glance

This section presents the configuration screen for BMX EHC 0200 counting modules.

Illustration

The figure below presents the configuration screen for the BMX EHC 0200 module in modulo loop counter mode:



NOTE: When adding a BMX EHC 0200 in a local rack the defaut function is **Frequency mode**

Description of the Screen

The following table presents the various parts of the above screen:

Number	Element	Function
1	Tab	The tab in the foreground indicates the current mode. The current mode is therefore the configuration mode in this example.
2	**Label** field	This field contains the name of each variable that may be configured. This field may not be modified.
3	**Symbol** field	This field contains the address of the variable in the application. This field may not be modified.
4	**Value** field	If this field has a downward pointing arrow, you can select the value of each variable from various possible values in this field. The various values can be accessed by clicking on the arrow. A drop-down menu containing all the possible values is displayed and the user may then select the required value of the variable.
5	**Unit** field	This field contains the unit of each variable that may be configured. This field may not be modified.

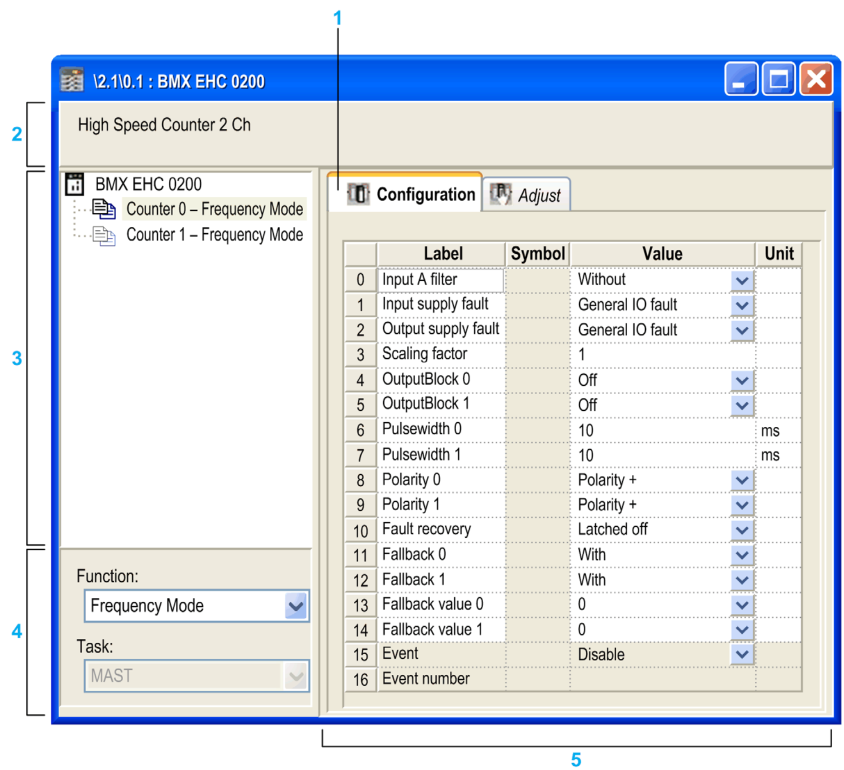
BMX EHC 0200 Counting Module Configuration Screens in X80 Drop

Introduction

The various available screens for the BMX EHC 0200 counting modules are:
- Configuration screen
- Adjust screen

Description of the Screens

The following diagram presents the counting modules configuration screen.



The following table presents the parts of the various screens.

Number	Element	Function
1	Tabs	The tab in the foreground indicates the mode in progress (**Configuration** in this example). Every mode can be selected using the respective tab. The available modes are: • **Configuration** • **Adjust**
2	**Module** area	Provides an abbreviation as a reminder of the module and module status in online mode (LEDs). Is used: • By clicking on the reference number, to display the tabs: • **Device DDT**
3	**Channel** area	Is used: • By clicking on the reference number, to display the tabs: • **Description** which gives the characteristics of the device. • To select a channel. • To display the **Symbol**, name of the channel defined by the user (using the variable editor). **NOTE:** All channel are activated and a channel can not be desactivated to **None**
4	**General parameters** area	Allows you to select the counting function and the task associated with the channel: • **Function:** counting function among those available for the modules involved. Depending on this choice, the headings of the configuration area may differ. By default, **Frequency Mode** no function is configured. • **Task:** defines the **MAST** task through which the channel's implicit exchange objects will be exchanged. These choices are only possible in offline mode.
5	**Parameters in progress** area	This area has various functionalities which depend upon the current mode: • **Configuration**: is used to configure the channel parameters. • **Adjust**: consists of various sections to be completed (parameter values), displayed according to the choice of counting function. **NOTE:** The Input and Output fault parameters are set by default with the value **Local** or **General IO Fault**.

9.2 Configuration of Modes for the BMX EHC 0200 Module

Subject of this Section

This section deals with the configuration of the modes of the BMX EHC 0200 counting modules.

What Is in This Section?

This section contains the following topics:

Frequency Mode Configuration

At a Glance

The configuration of a counting module is stored in the configuration constants (%KW).

The parameters r,m and c shown in the following tables represent the topologic addressing of the module. Each parameter had the following signification:

- r: represents the rack number,
- m:represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents the frequency mode configurable elements.

Label	Address in the configuration	Configurable values
Counting mode	%KWr.m.c.2 (least significant byte)	Frequency mode. The value of the least significant byte of this word is 1.
IN_A input filter	%KWr.m.c.3 (least significant byte)	The least significant byte can take the following values: • 0: none, • 1: low, • 2: medium, • 3: high.
Input power supply fault	%KWr.m.c.2.8	General input/output fault (bit set to 0) Local (bit set to 1)
Scale factor	%KWr.m.c.6 (least significant byte)	Edit (value in the range 1...255)
Output block 0	%KWr.m.c.17	This word can take the following values: • 0: off, • 1: low counter, • 2: counter in a window, • 3: High counter, • 4: pulse = less than the lower threshold (LT), • 5: pulse = greater than the lower threshold (LT), • 6: pulse = less than the upper threshold (UT), • 7: pulse = greater than the upper threshold (UT).

Label	Address in the configuration	Configurable values
Output block 1	%KWr.m.c.19	This word can take the following values: • 0: off, • 1: low counter, • 2: counter in a window, • 3: High counter, • 4: pulse = less than the lower threshold (LT), • 5: pulse = greater than the lower threshold (LT), • 6: pulse = less than the upper threshold (UT), • 7: pulse = greater than the upper threshold (UT).
Polarity 0	%KWr.m.c.21.1	Polarity + (bit set to 0) Polarity - (bit set to 1)
Polarity 1	%KWr.m.c.21.2	Polarity + (bit set to 0) Polarity - (bit set to 1)
Fault recovery	%KWr.m.c.21.0	Automatic reaction (bit set to 1) Activated (bit set to 0)
Fallback 0	%KWr.m.c.21.3	None (bit set to 0) With (bit set to 1)
Fallback 1	%KWr.m.c.21.4	None (bit set to 0) With (bit set to 1)
Fallback value 0	%KWr.m.c.21.5	0 (bit set to 0) 1 (bit set to 1)
Fallback value 1	%KWr.m.c.21.6	0 (bit set to 0) 1 (bit set to 1)
Output power supply fault	%KWr.m.c.2.9	General input/output fault (bit set to 0) Offline (bit set to 1)
Pulse width 0	%KWr.m.c.18	Edit (value in the range 1...65535)
Pulse width 1	%KWr.m.c.20	Edit (value in the range 1...65535)
Event Event number	%KWr.m.c.0	Activated (if activated is selected, the entered event number is coded on the most significant byte of this word) Deactivated (all bits of the most significant byte of this word are set to 1)

Event Counting Mode Configuration

At a Glance

The configuration of a counting module is stored in the configuration constants (%KW).

The parameters r,m and c shown in the following tables represent the topologic addressing of the module. Each parameter had the following signification:

- r: represents the rack number,
- m:represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents the event counting mode configurable elements

Label	Address in the configuration	Configurable values
Counting mode	%KWr.m.c.2 (least significant byte)	Event counting mode. The value of the least significant byte of this word is 2.
IN_A input filter	%KWr.m.c.3 (least significant byte)	The least significant byte can take the following values: • 0: none, • 1: low, • 2: medium, • 3: high.
IN_SYNC input filter	%KWr.m.c.4 (least significant byte)	The least significant byte can take the following values: • 0: none, • 1: low, • 2: medium, • 3: high.
Input power supply fault	%KWr.m.c.2.8	General input/output fault (bit set to 0) Local (bit set to 1)
Synchronization edge	%KWr.m.c.10.8	Rising edge at IN_SYNC (bit set to 0) Falling edge at IN_SYNC (bit set to 1)
Time base	%KWr.m.c.7	This word can take the following values: • 0: 0.1 s, • 1: 1 s, • 2: 10 s, • 3: 1 min

Label	Address in the configuration	Configurable values
Output block 0	`%KWr.m.c.17`	This word can take the following values: ● 0: off, ● 1: low counter, ● 2: counter in a window, ● 3: High counter, ● 4: pulse = less than the lower threshold (LT), ● 5: pulse = greater than the lower threshold (LT), ● 6: pulse = less than the upper threshold (UT), ● 7: pulse = greater than the upper threshold (UT).
Output block 1	`%KWr.m.c.19`	This word can take the following values: ● 0: off, ● 1: low counter, ● 2: counter in a window, ● 3: High counter, ● 4: pulse = less than the lower threshold (LT), ● 5: pulse = greater than the lower threshold (LT), ● 6: pulse = less than the upper threshold (UT), ● 7: pulse = greater than the upper threshold (UT).
Polarity 0	`%KWr.m.c.21.1`	Polarity + (bit set to 0) Polarity - (bit set to 1)
Polarity 1	`%KWr.m.c.21.2`	Polarity + (bit set to 0) Polarity - (bit set to 1)
Fault recovery	`%KWr.m.c.21.0`	Automatic reaction (bit set to 1) Activated (bit set to 0)
Fallback 0	`%KWr.m.c.21.3`	None (bit set to 0) With (bit set to 1)
Fallback 1	`%KWr.m.c.21.4`	None (bit set to 0) With (bit set to 1)
Fallback value 0	`%KWr.m.c.21.5`	0 (bit set to 0) 1 (bit set to 1)
Fallback value 1	`%KWr.m.c.21.6`	0 (bit set to 0) 1 (bit set to 1)
Output power supply fault	`%KWr.m.c.2.9`	General input/output fault (bit set to 0) Offline (bit set to 1)
Pulse width 0	`%KWr.m.c.18`	Edit (value in the range 1...65535)
Pulse width 1	`%KWr.m.c.20`	Edit (value in the range 1...65535)
Event Event number	`%KWr.m.c.0`	Activated (if activated is selected, the entered event number is coded on the most significant byte of this word) Deactivated (all bits of the most significant byte of this word are set to 1)

Period Measuring Mode Configuration

At a Glance

The configuration of a counting module is stored in the configuration constants (%KW).

The parameters r,m and c shown in the following tables represent the topologic addressing of the module. Each parameter had the following signification:

- r: represents the rack number,
- m:represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents the period measuring mode configurable elements.

Label	Address in the configuration	Configurable values
Counting mode	%KWr.m.c.2 (least significant byte)	Period measuring mode. The value of the least significant byte of this word is 3.
IN_A input filter	%KWr.m.c.3 (least significant byte)	The least significant byte can take the following values: ● 0: none, ● 1: low, ● 2: medium, ● 3: high.
IN_SYNC input filter	%KWr.m.c.4 (least significant byte)	The least significant byte can take the following values: ● 0: none, ● 1: low, ● 2: medium, ● 3: high.
Input power supply fault	%KWr.m.c.2.8	General input/output fault (bit set to 0) Local (bit set to 1)
Resolution	%KWr.m.c.8 (most significant byte)	The most significant byte can take the following values: ● 0: 1 µs, ● 1: 100 µs, ● 2: 1 ms.

Label	Address in the configuration	Configurable values
Mode	%KWr.m.c.8 (least significant byte)	The least significant byte can take the following values: ● 0: From one edge to the same edge at input IN_A, ● 1: From one edge to the opposite edge at input IN_A.
Time out	%KDr.m.c.14	0... 1 073 741 823
Output block 0	%KWr.m.c.17	This word can take the following values: ● 0: off, ● 1: low counter, ● 2: counter in a window, ● 3: High counter, ● 4: pulse = less than the lower threshold (LT), ● 5: pulse = greater than the lower threshold (LT), ● 6: pulse = less than the upper threshold (UT), ● 7: pulse = greater than the upper threshold (UT).
Output block 1	%KWr.m.c.19	This word can take the following values: ● 0: off, ● 1: low counter, ● 2: counter in a window, ● 3: High counter, ● 4: pulse = less than the lower threshold (LT), ● 5: pulse = greater than the lower threshold (LT), ● 6: pulse = less than the upper threshold (UT), ● 7: pulse = greater than the upper threshold (UT).
Polarity 0	%KWr.m.c.21.1	Polarity + (bit set to 0) Polarity - (bit set to 1)
Polarity 1	%KWr.m.c.21.2	Polarity + (bit set to 0) Polarity - (bit set to 1)
Fault recovery	%KWr.m.c.21.0	Automatic reaction (bit set to 1) Activated (bit set to 0)
Fallback 0	%KWr.m.c.21.3	None (bit set to 0) With (bit set to 1)
Fallback 1	%KWr.m.c.21.4	None (bit set to 0) With (bit set to 1)

Label	Address in the configuration	Configurable values
Fallback value 0	`%KWr.m.c.21.5`	0 (bit set to 0) 1 (bit set to 1)
Fallback value 1	`%KWr.m.c.21.6`	0 (bit set to 0) 1 (bit set to 1)
Output power supply fault	`%KWr.m.c.2.9`	General input/output fault (bit set to 0) Offline (bit set to 1)
Pulse width 0	`%KWr.m.c.18`	Edit (value in the range 1...65535)
Pulse width 1	`%KWr.m.c.20`	Edit (value in the range 1...65535)
Event Event number	`%KWr.m.c.0`	Activated (if activated is selected, the entered event number is coded on the most significant byte of this word) Deactivated (all bits of the most significant byte of this word are set to 1)

Ratio Mode Configuration

At a Glance

The configuration of a counting module is stored in the configuration constants (%KW).

The parameters r,m and c shown in the following tables represent the topologic addressing of the module. Each parameter had the following signification:

- r: represents the rack number,
- m:represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents ratio mode configurable elements.

Label	Address in the configuration	Configurable values
Counting mode	%KWr.m.c.2 (least significant byte)	The least significant byte of this word can take the following values in this mode: ● 4: ratio 1 mode, ● 5: ratio 2 mode.
IN_A input filter	%KWr.m.c.3 (least significant byte)	The least significant byte can take the following values: ● 0: none, ● 1: low, ● 2: medium, ● 3: high.
IN_B input filter	%KWr.m.c.3 (most significant byte)	The most significant byte can take the following values: ● 0: none, ● 1: low, ● 2: medium, ● 3: high.
Input power supply fault	%KWr.m.c.2.8	General input/output fault (bit set to 0) Local (bit set to 1)
Scale factor	%KWr.m.c.6 (least significant byte)	Edit (value in the range 1...255)
Absolute limit	%KDr.m.c.12	Edit

Label	Address in the configuration	Configurable values
Output block 0	`%KWr.m.c.17`	This word can take the following values: ● 0: off, ● 1: low counter, ● 2: counter in a window, ● 3: High counter, ● 4: pulse = less than the lower threshold (LT), ● 5: pulse = greater than the lower threshold (LT), ● 6: pulse = less than the upper threshold (UT), ● 7: pulse = greater than the upper threshold (UT).
Output block 1	`%KWr.m.c.19`	This word can take the following values: ● 0: off, ● 1: low counter, ● 2: counter in a window, ● 3: High counter, ● 4: pulse = less than the lower threshold (LT), ● 5: pulse = greater than the lower threshold (LT), ● 6: pulse = less than the upper threshold (UT), ● 7: pulse = greater than the upper threshold (UT).
Polarity 0	`%KWr.m.c.21.1`	Polarity + (bit set to 0) Polarity - (bit set to 1)
Polarity 1	`%KWr.m.c.21.2`	Polarity + (bit set to 0) Polarity - (bit set to 1)
Fault recovery	`%KWr.m.c.21.0`	Automatic reaction (bit set to 1) Activated (bit set to 0)
Fallback 0	`%KWr.m.c.21.3`	None (bit set to 0) With (bit set to 1)
Fallback 1	`%KWr.m.c.21.4`	None (bit set to 0) With (bit set to 1)
Fallback value 0	`%KWr.m.c.21.5`	0 (bit set to 0) 1 (bit set to 1)
Fallback value 1	`%KWr.m.c.21.6`	0 (bit set to 0) 1 (bit set to 1)
Output power supply fault	`%KWr.m.c.2.9`	General input/output fault (bit set to 0) Offline (bit set to 1)
Pulse width 0	`%KWr.m.c.18`	Edit (value in the range 1...65535)
Pulse width 1	`%KWr.m.c.20`	Edit (value in the range 1...65535)
Event Event number	`%KWr.m.c.0`	Activated (if activated is selected, the entered event number is coded on the most significant byte of this word) Deactivated (all bits of the most significant byte of this word are set to 1)

One Shot Counter Mode Configuration

At a Glance

The configuration of a counting module is stored in the configuration constants (%KW).

The parameters r,m and c shown in the following tables represent the topologic addressing of the module. Each parameter had the following signification:

- r: represents the rack number,
- m:represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents the one shot counter mode configurable elements

Label	Address in the configuration	Configurable values
Counting mode	%KWr.m.c.2 (least significant byte)	One shot counter mode. The value of the least significant byte of this word is 6.
IN_A input filter	%KWr.m.c.3 (least significant byte)	The least significant byte can take the following values: ● 0: none, ● 1: low, ● 2: medium, ● 3: high.
IN_SYNC input filter	%KWr.m.c.4 (least significant byte)	The least significant byte can take the following values: ● 0: none, ● 1: low, ● 2: medium, ● 3: high.
IN_EN input filter	%KWr.m.c.4 (most significant byte)	The most significant byte can take the following values: ● 0: none, ● 1: low, ● 2: medium, ● 3: high.
Input power supply fault	%KWr.m.c.2.8	General input/output fault (bit set to 0) Local (bit set to 1)
Scale factor	%KWr.m.c.6 (least significant byte)	Edit (value in the range 1...255)

Label	Address in the configuration	Configurable values
Synchronization edge	`%KWr.m.c.10.8`	Rising edge (bit set to 0) Falling edge (bit set to 1)
Output block 0	`%KWr.m.c.17`	This word can take the following values: ● 0: off, ● 1: low counter, ● 2: counter in a window, ● 3: High counter, ● 4: pulse = less than the lower threshold (LT), ● 5: pulse = greater than the lower threshold (LT), ● 6: pulse = less than the upper threshold (UT), ● 7: pulse = greater than the upper threshold (UT).
Output block 1	`%KWr.m.c.19`	This word can take the following values: ● 0: off, ● 1: low counter, ● 2: counter in a window, ● 3: High counter, ● 4: pulse = less than the lower threshold (LT), ● 5: pulse = greater than the lower threshold (LT), ● 6: pulse = less than the upper threshold (UT), ● 7: pulse = greater than the upper threshold (UT).
Polarity 0	`%KWr.m.c.21.1`	Polarity + (bit set to 0) Polarity - (bit set to 1)
Polarity 1	`%KWr.m.c.21.2`	Polarity + (bit set to 0) Polarity - (bit set to 1)
Fault recovery	`%KWr.m.c.21.0`	Automatic reaction (bit set to 1) Activated (bit set to 0)
Fallback 0	`%KWr.m.c.21.3`	None (bit set to 0) With (bit set to 1)
Fallback 1	`%KWr.m.c.21.4`	None (bit set to 0) With (bit set to 1)
Fallback value 0	`%KWr.m.c.21.5`	0 (bit set to 0) 1 (bit set to 1)
Fallback value 1	`%KWr.m.c.21.6`	0 (bit set to 0) 1 (bit set to 1)
Output power supply fault	`%KWr.m.c.2.9`	General input/output fault (bit set to 0) Offline (bit set to 1)

Label	Address in the configuration	Configurable values
Pulse width 0	`%KWr.m.c.18`	Edit (value in the range 1...65535)
Pulse width 1	`%KWr.m.c.20`	Edit (value in the range 1...65535)
Event Event number	`%KWr.m.c.0`	Activated (if activated is selected, the entered event number is coded on the most significant byte of this word) Deactivated (all bits of the most significant byte of this word are set to 1)

Modulo Loop Counter Mode Configuration

At a Glance

The configuration of a counting module is stored in the configuration constants (%KW).

The parameters r,m and c shown in the following tables represent the topologic addressing of the module. Each parameter had the following signification:

- r: represents the rack number,
- m:represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents modulo loop counter mode configurable elements.

Label	Address in the configuration	Configurable values
Counting mode	%KWr.m.c.2 (least significant byte)	Modulo loop counter mode. The value of the least significant byte of this word is 7.
IN_A input filter	%KWr.m.c.3 (least significant byte)	The least significant byte can take the following values: • 0: none, • 1: low, • 2: medium, • 3: high.
IN_A input filter	%KWr.m.c.3 (most significant byte)	The most significant byte can take the following values: • 0: none, • 1: low, • 2: medium, • 3: high.
IN_SYNC input filter	%KWr.m.c.4 (least significant byte)	The least significant byte can take the following values: • 0: none, • 1: low, • 2: medium, • 3: high.
IN_EN input filter	%KWr.m.c.4 (most significant byte)	The most significant byte can take the following values: • 0: none, • 1: low, • 2: medium, • 3: high.

Label	Address in the configuration	Configurable values
Input power supply fault	%KWr.m.c.2.8	General input/output fault (bit set to 0) Local (bit set to 1)
Input mode	%KWr.m.c.9	This word can take the following values: ● 0: A = High, B = Low, ● 1: A = Pulse, B = Direction, ● 2: normal quadrature 1, ● 3: normal quadrature 2, ● 4: normal quadrature 4, ● 5: inverse quadrature 1, ● 6: inverse quadrature 2, ● 7: inverse quadrature 4.
Scale factor	%KWr.m.c.6 (least significant byte)	Edit (value in the range 1...255)
Synchronization edge	%KWr.m.c.10 (most significant byte)	Rising edge (bit set to 0) Falling edge (bit set to 1)
Output block 0	%KWr.m.c.17	This word can take the following values: ● 0: off, ● 1: low counter, ● 2: counter in a window, ● 3: High counter, ● 4: pulse = less than the lower threshold (LT), ● 5: pulse = greater than the lower threshold (LT), ● 6: pulse = less than the upper threshold (UT), ● 7: pulse = greater than the upper threshold (UT).
Output block 1	%KWr.m.c.19	This word can take the following values: ● 0: off, ● 1: low counter, ● 2: counter in a window, ● 3: High counter, ● 4: pulse = less than the lower threshold (LT), ● 5: pulse = greater than the lower threshold (LT), ● 6: pulse = less than the upper threshold (UT), ● 7: pulse = greater than the upper threshold (UT).

Label	Address in the configuration	Configurable values
Polarity 0	%KWr.m.c.21.1	Polarity + (bit set to 0) Polarity - (bit set to 1)
Polarity 1	%KWr.m.c.21.2	Polarity + (bit set to 0) Polarity - (bit set to 1)
Fault recovery	%KWr.m.c.21.0	Automatic reaction (bit set to 1) Activated (bit set to 0)
Fallback 0	%KWr.m.c.21.3	None (bit set to 0) With (bit set to 1)
Fallback 1	%KWr.m.c.21.4	None (bit set to 0) With (bit set to 1)
Fallback value 0	%KWr.m.c.21.5	0 (bit set to 0) 1 (bit set to 1)
Fallback value 1	%KWr.m.c.21.6	0 (bit set to 0) 1 (bit set to 1)
Output power supply fault	%KWr.m.c.2.9	General input/output fault (bit set to 0) Offline (bit set to 1)
Pulse width 0	%KWr.m.c.18	Edit (value in the range 1...65535)
Pulse width 1	%KWr.m.c.20	Edit (value in the range 1...65535)
Event Event number	%KWr.m.c.0	Activated (if activated is selected, the entered event number is coded on the most significant byte of this word) Deactivated (all bits of the most significant byte of this word are set to 1)

Free Large Counter Mode Configuration

At a Glance

The configuration of a counting module is stored in the configuration constants (%KW).

The parameters r,m and c shown in the following tables represent the topologic addressing of the module. Each parameter had the following signification:

- r: represents the rack number,
- m:represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents the free large counter mode configurable elements.

Label	Address in the configuration	Configurable values
Counting mode	%KWr.m.c.2 (least significant byte)	Free large counter mode. The value of the least significant byte of this word is 8.
IN_A input filter	%KWr.m.c.3 (least significant byte)	The least significant byte can take the following values: ● 0: none, ● 1: low, ● 2: medium, ● 3: high.
IN_B input filter	%KWr.m.c.3 (most significant byte)	The most significant byte can take the following values: ● 0: none, ● 1: low, ● 2: medium, ● 3: high.
IN_SYNC input filter	%KWr.m.c.4 (least significant byte)	The least significant byte can take the following values: ● 0: none, ● 1: low, ● 2: medium, ● 3: high.
IN_EN input filter	%KWr.m.c.4 (most significant byte)	The most significant byte can take the following values: ● 0: none, ● 1: low, ● 2: medium, ● 3: high.

Label	Address in the configuration	Configurable values
IN_REF input filter	`%KWr.m.c.5` (least significant byte)	The least significant byte can take the following values: ● 0: none, ● 1: low, ● 2: medium, ● 3: high.
IN_CAP input filter	`%KWr.m.c.5` (most significant byte)	The most significant byte can take the following values: ● 0: none, ● 1: low, ● 2: medium, ● 3: high.
Input power supply fault	`%KWr.m.c.2.8`	General input/output fault (bit set to 0) Local (bit set to 1)
Input mode	`%KWr.m.c.9`	This word can take the following values: ● 0: A = High, B = Low ● 1: A = Pulse, B = Direction ● 2: normal quadrature 1 ● 3: normal quadrature 2 ● 4: normal quadrature 4 ● 5: inverse quadrature 1 ● 6: inverse quadrature 2 ● 7: inverse quadrature 4
Scale factor	`%KWr.m.c.6` (least significant byte)	Edit (value in the range 1...255)
Preset mode	`%KWr.m.c.10` (least significant byte)	The least significant byte can take the following values: ● 0: rising edge at IN_SYNC ● 1: rising edge at IN_REF ● 2: rising edge at IN_SYNC and IN_REF ● 3: first rising edge at IN_SYNC and IN_REF at 1 ● 4: first rising edge at IN_SYNC and IN_REF at 0
Capture 0 settings	`%KWr.m.c.16.1`	Preset condition (bit set to 0) Falling edge at the IN_CAP input (bit set to 1)

Label	Address in the configuration	Configurable values
Output block 0	`%KWr.m.c.17`	This word can take the following values: • 0: off, • 1: low counter, • 2: counter in a window, • 3: High counter, • 4: pulse = less than the lower threshold (LT), • 5: pulse = greater than the lower threshold (LT), • 6: pulse = less than the upper threshold (UT), • 7: pulse = greater than the upper threshold (UT).
Output block 1	`%KWr.m.c.19`	This word can take the following values: • 0: off, • 1: low counter, • 2: counter in a window, • 3: High counter, • 4: pulse = less than the lower threshold (LT), • 5: pulse = greater than the lower threshold (LT), • 6: pulse = less than the upper threshold (UT), • 7: pulse = greater than the upper threshold (UT).
Polarity 0	`%KWr.m.c.21.1`	Polarity + (bit set to 0) Polarity - (bit set to 1)
Polarity 1	`%KWr.m.c.21.2`	Polarity + (bit set to 0) Polarity - (bit set to 1)
Fault recovery	`%KWr.m.c.21.0`	Automatic reaction (bit set to 1) Activated (bit set to 0)
Fallback 0	`%KWr.m.c.21.3`	None (bit set to 0) With (bit set to 1)
Fallback 1	`%KWr.m.c.21.4`	None (bit set to 0) With (bit set to 1)
Fallback value 0	`%KWr.m.c.21.5`	0 (bit set to 0) 1 (bit set to 1)
Fallback value 1	`%KWr.m.c.21.6`	0 (bit set to 0) 1 (bit set to 1)
Output power supply fault	`%KWr.m.c.2.9`	General input/output fault (bit set to 0) Offline (bit set to 1)

Label	Address in the configuration	Configurable values
Pulse width 0	%KWr.m.c.18	Edit (value in the range 1...65535)
Pulse width 1	%KWr.m.c.20	Edit (value in the range 1...65535)
Event Event number	%KWr.m.c.0	Activated (if activated is selected, the entered event number is coded on the most significant byte of this word) Deactivated (all bits of the most significant byte of this word are set to 1)

Pulse Width Modulation Mode Configuration

At a Glance

The configuration of a counting module is stored in the configuration constants (%KW).

The parameters r,m and c shown in the following tables represent the topologic addressing of the module. Each parameter had the following signification:

- r: represents the rack number,
- m:represents the position of the module on the rack,
- c: represents the channel number.

Configuration Objects

The table below presents the pulse width modulation mode configurable elements.

Label	Address in the configuration	Configurable values
Counting mode	%KWr.m.c.2 (least significant byte)	Pulse width modulation mode. The value of the least significant byte of this word is 9.
IN_SYNC input filter	%KWr.m.c.4 (least significant byte)	The least significant byte can take the following values: ● 0: none, ● 1: low, ● 2: medium, ● 3: high.
Synchronization edge	%KWr.m.c.10.8	Rising edge at IN_SYNC (bit set to 0) Falling edge at IN_SYNC (bit set to 1)
IN_EN input filter	%KWr.m.c.4 (most significant byte)	The most significant byte can take the following values: ● 0: none, ● 1: low, ● 2: medium, ● 3: high.
Input power supply fault	%KWr.m.c.2.8	General input/output fault (bit set to 0) Local (bit set to 1)
Polarity 0	%KWr.m.c.21.1	Polarity + (bit set to 0) Polarity - (bit set to 1)
Polarity 1	%KWr.m.c.21.2	Polarity + (bit set to 0) Polarity - (bit set to 1)
Fault recovery	%KWr.m.c.21.0	Automatic reaction (bit set to 1) Activated (bit set to 0)

Label	Address in the configuration	Configurable values
Fallback 0	%KWr.m.c.21.3	None (bit set to 0) With (bit set to 1)
Fallback 1	%KWr.m.c.21.4	None (bit set to 0) With (bit set to 1)
Fallback value 0	%KWr.m.c.21.5	0 (bit set to 0) 1 (bit set to 1)
Fallback value 1	%KWr.m.c.21.6	0 (bit set to 0) 1 (bit set to 1)
Output power supply fault	%KWr.m.c.2.9	General input/output fault (bit set to 0) Offline (bit set to 1)
Event Event number	%KWr.m.c.0	Activated (if activated is selected, the entered event number is coded on the most significant byte of this word) Deactivated (all bits of the most significant byte of this word are set to 1)

BMX EHC xxxx Counting Module Settings

10

Subject of this Chapter

This chapter deals with the possible settings for the counting modes of the BMX EHC •••• modules. These settings can be accessed from the Configuration tab on the functional screens of BMX EHC •••• modules *(see page 108)*.

What Is in This Chapter?

This chapter contains the following topics:

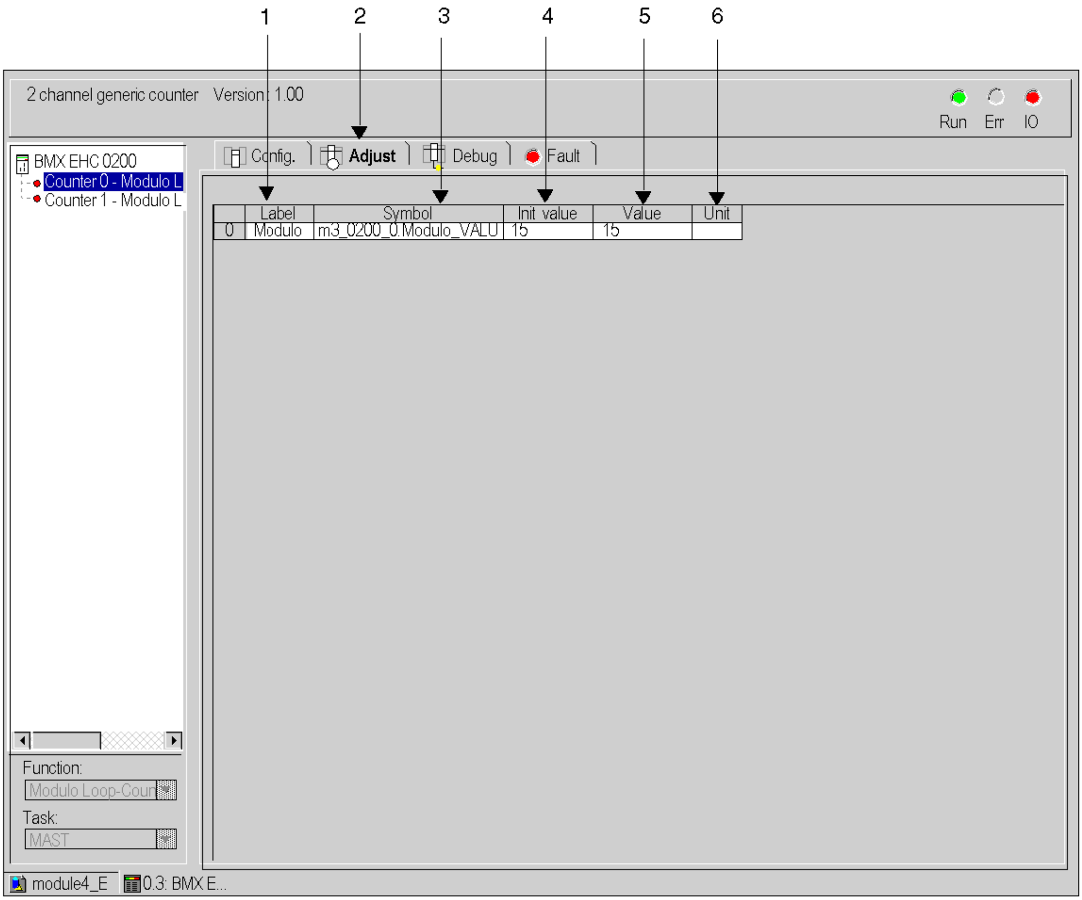
Adjust Screen for BMX EHC 0200 Counting Modules

At a Glance

This section presents the adjust screen for BMX EHC 0200 counting modules.

Illustration

The figure below presents the adjust screen for the BMX EHC 0200 module in modulo loop counter mode:



Description of the Screen

The following table presents the various parts of the above screen:

Number	Element	Function
1	**Label** field	This field contains the name of each variable that may be adjusted. This field may not be modified and can be accessed in both local and online modes.
2	Tab	The tab in the foreground indicates the current mode. The current mode is therefore the adjust mode in this example.
3	**Symbol** field	This field contains the mnemonics of the variable. This field may not be modified and can be accessed in both offline and online modes.
4	**Initial value** field	This field displays the value of the variable that the user has adjusted in offline mode. This field is only accessible in online mode.
5	**Value** field	The function of this field depends on the mode in which the user is working: ● **In offline mode**: this field is used to adjust the variable. ● **In online mode**: this field is used to display the current value of the variable.
6	**Unit** field	This field contains the unit of each variable that may be configured. This field may not be modified and can be accessed in both offline and online modes.

Setting the Preset Value

Introduction

The preset value concerns the following counting modes:

- for the BMX EHC 0200 module:
  - one shot counter mode
  - free large counter mode

Description

The following table shows the preset value setting:

Number	Address in the configuration	Value	Default value
Preset value	%MDr.m.c.12 (Low)	Edit	0

Setting the Calibration Factor

Introduction

The calibration factor concerns the frequency and ratio modes for the BMX EHC 0200 .

Description

The following table shows the calibration factor setting:

Number	Address in the configuration	Value	Default value
Calibration factor	%MWr.m.c.14	Edit	0

Modulo Adjust

Introduction

The modulo concerns the modulo loop counter modes for the counting modules BMX EHC ****.

Description

The following table shows the modulo adjust:

Number	Address in the configuration	Value	Default value
Modulo	`%MDx.y.v.10` (Low)	Edit	0xFFFF

Setting the Hysteresis Value

Introduction

The hysteresis value concerns free large counter mode for module BMX EHC 0200.

Description

The following table shows the setting for the hysteresis value:

Number	Address in the configuration	Value	Default value
Hysteresis (release value)	`%MWr.m.c.9`	Edit	0

Debugging the BMX EHC 0200 Counting Modules

11

Subject of this Chapter

This chapter deals with the debugging settings applicable to BMX EHC 0200 modules. These settings can be accessed from the Debug tab on the functional screens of the BMX EHC 0200 *(see page 106)* modules.

What Is in This Chapter?

This chapter contains the following sections:

11.1 Debug Screen for BMX EHC xxxx Counting Modules

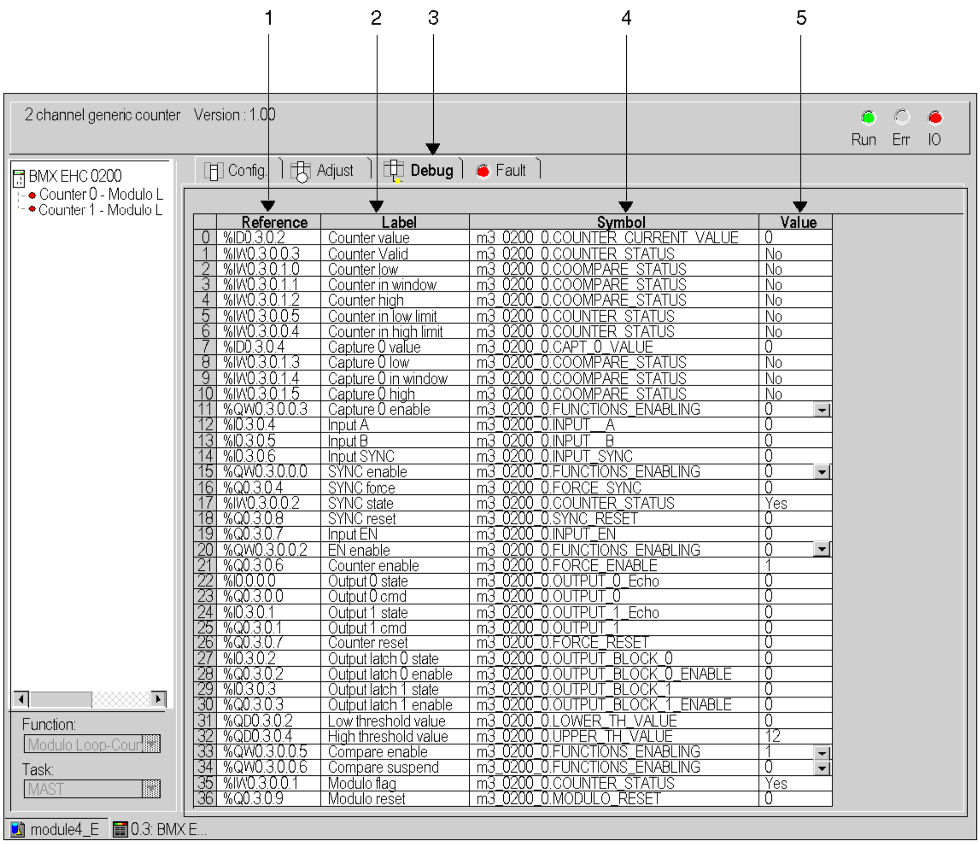
Debug Screen for BMX EHC xxxx Counting Modules

At a Glance

This section presents the debug screen for BMX EHC •••• counting modules. A module’s debug screen can only be accessed in online mode.

Illustration

The figure below presents the debug screen for the BMX EHC 0200 module in modulo loop counter mode:



	Reference	Label	Symbol	Value
0	%ID0.3.0.2	Counter value	m3_0200_0.COUNTER_CURRENT_VALUE	0
1	%IW0.3.0.0.3	Counter Valid	m3_0200_0.COUNTER_STATUS	No
2	%IW0.3.0.1.0	Counter low	m3_0200_0.COOMPARE_STATUS	No
3	%IW0.3.0.1.1	Counter in window	m3_0200_0.COOMPARE_STATUS	No
4	%IW0.3.0.1.2	Counter high	m3_0200_0.COOMPARE_STATUS	No
5	%IW0.3.0.0.5	Counter in low limit	m3_0200_0.COUNTER_STATUS	No
6	%IW0.3.0.0.4	Counter in high limit	m3_0200_0.COUNTER_STATUS	No
7	%ID0.3.0.4	Capture 0 value	m3_0200_0.CAPT_0_VALUE	0
8	%IW0.3.0.1.3	Capture 0 low	m3_0200_0.COOMPARE_STATUS	No
9	%IW0.3.0.1.4	Capture 0 in window	m3_0200_0.COOMPARE_STATUS	No
10	%IW0.3.0.1.5	Capture 0 high	m3_0200_0.COOMPARE_STATUS	No
11	%QW0.3.0.0.3	Capture 0 enable	m3_0200_0.FUNCTIONS_ENABLING	0
12	%I0.3.0.4	Input A	m3_0200_0.INPUT__A	0
13	%I0.3.0.5	Input B	m3_0200_0.INPUT__B	0
14	%I0.3.0.6	Input SYNC	m3_0200_0.INPUT_SYNC	0
15	%QW0.3.0.0.0	SYNC enable	m3_0200_0.FUNCTIONS_ENABLING	0
16	%Q0.3.0.4	SYNC force	m3_0200_0.FORCE_SYNC	0
17	%IW0.3.0.0.2	SYNC state	m3_0200_0.COUNTER_STATUS	Yes
18	%Q0.3.0.8	SYNC reset	m3_0200_0.SYNC_RESET	0
19	%Q0.3.0.7	Input EN	m3_0200_0.INPUT_EN	0
20	%QW0.3.0.0.2	EN enable	m3_0200_0.FUNCTIONS_ENABLING	0
21	%Q0.3.0.6	Counter enable	m3_0200_0.FORCE_ENABLE	1
22	%I0.0.0.0	Output 0 state	m3_0200_0.OUTPUT_0_Echo	0
23	%Q0.3.0.0	Output 0 cmd	m3_0200_0.OUTPUT_0	0
24	%I0.3.0.1	Output 1 state	m3_0200_0.OUTPUT_1_Echo	0
25	%Q0.3.0.1	Output 1 cmd	m3_0200_0.OUTPUT_1	0
26	%Q0.3.0.7	Counter reset	m3_0200_0.FORCE_RESET	0
27	%I0.3.0.2	Output latch 0 state	m3_0200_0.OUTPUT_BLOCK_0	0
28	%Q0.3.0.2	Output latch 0 enable	m3_0200_0.OUTPUT_BLOCK_0_ENABLE	0
29	%I0.3.0.3	Output latch 1 state	m3_0200_0.OUTPUT_BLOCK_1	0
30	%Q0.3.0.3	Output latch 1 enable	m3_0200_0.OUTPUT_BLOCK_1_ENABLE	0
31	%QD0.3.0.2	Low threshold value	m3_0200_0.LOWER_TH_VALUE	0
32	%QD0.3.0.4	High threshold value	m3_0200_0.UPPER_TH_VALUE	12
33	%QW0.3.0.0.5	Compare enable	m3_0200_0.FUNCTIONS_ENABLING	1
34	%QW0.3.0.0.6	Compare suspend	m3_0200_0.FUNCTIONS_ENABLING	0
35	%IW0.3.0.0.1	Modulo flag	m3_0200_0.COUNTER_STATUS	Yes
36	%Q0.3.0.9	Modulo reset	m3_0200_0.MODULO_RESET	0

Description of the Screen

The following table presents the various parts of the above screen:

Number	Element	Function
1	**Reference** field	This field contains the address of the variable in the application. This field may not be modified.
2	**Label** field	This field contains the name of each variable that may be configured. This field may not be modified.
3	Tab	The tab in the foreground indicates the current mode. The current mode is therefore the debug mode in this example.
4	**Symbol** field	This field contains the mnemonics of the variable. This field may not be modified.
5	**Value** field	If the field has a downward pointing arrow, you can select the value of each variable from various possible values in this field. The various values can be accessed by clicking on the arrow. A drop-down menu containing all the possible values is displayed and the user may then select the required value of the variable. If there is no downward pointing arrow, this field simply displays the current value of the variable.

11.2 BMX EHC 0200 Module Debugging

Subject of this Section

This section deals with the debugging of the BMX EHC 0200 counting module modes.

What Is in This Section?

This section contains the following topics:

Frequency Mode Debugging

At a Glance

The table below presents the frequency mode debugging elements:

Label	Language object	Type
Frequency value	%IDr.m.c.2	Digital
Frequency valid	%IWr.m.c.0.3	Binary
Frequency low	%IWr.m.c.1.0	Binary
Frequency in window	%IWr.m.c.1.1	Binary
Frequency high	%IWr.m.c.1.2	Binary
Frequency in high limit	%IWr.m.c.0.4	Binary
Input A state	%Ir.m.c.4	Binary
Output 0 state	%Ir.m.c.0	Binary
Output 0 cmd	%Qr.m.c.0	Binary
Output 1 state	%Ir.m.c.1	Binary
Output 1 cmd	%Qr.m.c.1	Binary
Output latch 0 state	%Ir.m.c.2	Binary
Output latch 0 enable	%Qr.m.c.2	Binary
Output latch 1 state	%Ir.m.c.3	Binary
Output latch 1 enable	%Qr.m.c.3	Binary
Low threshold value	%QDr.m.c.2	Digital
High threshold value	%QDr.m.c.4	Digital
Compare enable	%QWr.m.c.0.5	Binary
Compare suspend	%QWr.m.c.0.6	Binary

For a description of each language object refer to T_UNSIGNED_CPT_BMX IODDT *(see page 180)*.

Event Counting Mode Debugging

At a Glance

The table below presents the event counting mode debugging elements:

Label	Language object	Type
Counter value	%IDr.m.c.2	Digital
Counter valid	%IWr.m.c.0.3	Binary
Counter low	%IWr.m.c.1.0	Binary
Counter in window	%IWr.m.c.1.1	Binary
Counter high	%IWr.m.c.1.2	Binary
Counter in low limit	%IWr.m.c.0.5	Binary
Counter in high limit	%IWr.m.c.0.4	Binary
Input A state	%Ir.m.c.4	Binary
Input SYNC state	%Ir.m.c.6	Binary
SYNC enable	%QWr.m.c.0.0	Binary
SYNC force	%Qr.m.c.4	Binary
SYNC state	%IWr.m.c.0.2	Binary
SYNC reset	%Qr.m.c.8	Binary
Output 0 state	%Ir.m.c.0	Binary
Output 0 cmd	%Qr.m.c.0	Binary
Output 1 state	%Ir.m.c.1	Binary
Output 1 cmd	%Qr.m.c.1	Binary
Output latch 0 state	%Ir.m.c.2	Binary
Output latch 0 enable	%Qr.m.c.2	Binary
Output latch 1 state	%Ir.m.c.3	Binary
Output latch 1 enable	%Qr.m.c.3	Binary
Low threshold value	%QDr.m.c.2	Digital
High threshold value	%QDr.m.c.4	Digital
Compare enable	%QWr.m.c.0.5	Binary
Compare suspend	%QWr.m.c.0.6	Binary

For a description of each language object refer to T_UNSIGNED_CPT_BMX IODDT *(see page 180)*.

Period Measuring Mode Debugging

At a Glance

The table below presents the period measuring mode debugging elements:

Label	Language object	Type
Period value	%IDr.m.c.2	Digital
Period valid	%IWr.m.c.0.3	Binary
Period low	%IWr.m.c.1.0	Binary
Period in window	%IWr.m.c.1.1	Binary
Period high	%IWr.m.c.1.2	Binary
Period in low limit	%IWr.m.c.0.5	Binary
Period in high limit	%IWr.m.c.0.4	Binary
Input A state	%Ir.m.c.4	Binary
Input SYNC state	%Ir.m.c.6	Binary
SYNC enable	%QWr.m.c.0.0	Binary
SYNC force	%Qr.m.c.4	Binary
SYNC state	%IWr.m.c.0.2	Binary
SYNC reset	%Qr.m.c.8	Binary
Output 0 state	%Ir.m.c.0	Binary
Output 0 cmd	%Qr.m.c.0	Binary
Output 1 state	%Ir.m.c.1	Binary
Output 1 cmd	%Qr.m.c.1	Binary
Output latch 0 state	%Ir.m.c.2	Binary
Output latch 0 enable	%Qr.m.c.2	Binary
Output latch 1 state	%Ir.m.c.3	Binary
Output latch 1 enable	%Qr.m.c.3	Binary
Low threshold value	%QDr.m.c.2	Digital
High threshold value	%QDr.m.c.4	Digital
Compare enable	%QWr.m.c.0.5	Binary
Compare suspend	%QWr.m.c.0.6	Binary

For a description of each language object refer to T_UNSIGNED_CPT_BMX IODDT *(see page 180)*.

Ratio Mode Debugging

At a Glance

The table below presents the ratio mode debugging elements:

Label	Language object	Type
Ratio value	%IDr.m.c.2	Digital
Ratio valid	%IWr.m.c.0.3	Binary
Ratio low	%IWr.m.c.1.0	Binary
Ratio in window	%IWr.m.c.1.1	Binary
Ratio high	%IWr.m.c.1.2	Binary
Ratio in low limit	%IWr.m.c.0.5	Binary
Ratio in high limit	%IWr.m.c.0.4	Binary
Input A state	%Ir.m.c.4	Binary
Input B state	%Ir.m.c.5	Binary
Output 0 state	%Ir.m.c.0	Binary
Output 0 cmd	%Qr.m.c.0	Binary
Output 1 state	%Ir.m.c.1	Binary
Output 1 cmd	%Qr.m.c.1	Binary
Output latch 0 state	%Ir.m.c.2	Binary
Output latch 0 enable	%Qr.m.c.2	Binary
Output latch 1 state	%Ir.m.c.3	Binary
Output latch 1 enable	%Qr.m.c.3	Binary
Low threshold value	%QDr.m.c.2	Digital
High threshold value	%QDr.m.c.4	Digital
Compare enable	%QWr.m.c.0.5	Binary
Compare suspend	%QWr.m.c.0.6	Binary

For a description of each language object refer to T_SIGNED_CPT_BMX IODDT *(see page 180)*.

One Shot Counter Mode Debugging

At a Glance

The table below presents the one shot counter mode debugging elements:

Label	Language object	Type
Counter value	%IDr.m.c.2	Digital
Counter valid	%IWr.m.c.0.3	Binary
Counter low	%IWr.m.c.1.0	Binary
Counter in window	%IWr.m.c.1.1	Binary
Counter high	%IWr.m.c.1.2	Binary
RUN	%IWr.m.c.0.0	Binary
Input A state	%Ir.m.c.4	Binary
Input SYNC state	%Ir.m.c.6	Binary
SYNC enable	%QWr.m.c.0.0	Binary
SYNC force	%Qr.m.c.4	Binary
SYNC state	%IWr.m.c.0.2	Binary
SYNC reset	%Qr.m.c.8	Binary
Input EN	%Ir.m.c.7	Binary
EN enable	%QWr.m.c.0.2	Binary
Counter enable	%Qr.m.c.6	Binary
Output 0 state	%Ir.m.c.0	Binary
Output 0 cmd	%Qr.m.c.0	Binary
Output 1 state	%Ir.m.c.1	Binary
Output 1 cmd	%Qr.m.c.1	Binary
Output latch 0 state	%Ir.m.c.2	Binary
Output latch 0 enable	%Qr.m.c.2	Binary
Output latch 1 state	%Ir.m.c.3	Binary
Output latch 1 enable	%Qr.m.c.3	Binary
Low threshold value	%QDr.m.c.2	Digital
High threshold value	%QDr.m.c.4	Digital
Compare enable	%QWr.m.c.0.5	Binary
Compare suspend	%QWr.m.c.0.6	Binary

For a description of each language object refer to T_UNSIGNED_CPT_BMX IODDT *(see page 180)*.

Modulo Loop Counter Mode Debugging

At a Glance

The table below presents the modulo loop counter mode debugging elements:

Label	Language object	Type
Counter value	%IDr.m.c.2	Digital
Counter valid	%IWr.m.c.0.3	Binary
Counter low	%IWr.m.c.1.0	Binary
Counter in window	%IWr.m.c.1.1	Binary
Counter high	%IWr.m.c.1.2	Binary
Counter in low limit	%IWr.m.c.0.5	Binary
Counter in high limit	%IWr.m.c.0.4	Binary
Capture value	%IDr.m.c.4	Digital
Capture low	%IWr.m.c.1.3	Binary
Capture in window	%IWr.m.c.1.4	Binary
Capture high	%IWr.m.c.1.5	Binary
Capture enable	%QWr.m.c.0.3	Binary
Input A state	%Ir.m.c.4	Binary
Input B state	%Ir.m.c.5	Binary
Input SYNC state	%Ir.m.c.6	Binary
SYNC enable	%QWr.m.c.0.0	Binary
SYNC force	%Qr.m.c.4	Binary
SYNC state	%IWr.m.c.0.2	Binary
SYNC reset	%QWr.m.c.8	Binary
Input EN	%Ir.m.c.7	Binary
EN enable	%QWr.m.c.0.2	Binary
Counter enable	%Qr.m.c.6	Binary
Output 0 state	%Ir.m.c.0	Binary
Output 0 cmd	%Qr.m.c.0	Binary
Output 1 state	%Ir.m.c.1	Binary
Output 1 cmd	%Qr.m.c.1	Binary
Counter reset	%Qr.m.c.7	Binary
Output latch 0 state	%Ir.m.c.2	Binary
Output latch 0 enable	%Qr.m.c.2	Binary
Output latch 1 state	%Ir.m.c.3	Binary

Label	Language object	Type
Output latch 01enable	%Qr.m.c.3	Binary
Low threshold value	%QDr.m.c.2	Digital
High threshold value	%QDr.m.c.4	Digital
Compare enable	%QWr.m.c.0.5	Binary
Compare suspend	%QWr.m.c.0.6	Binary
Modulo state	%IWr.m.c.0.1	Binary
Modulo reset	%Qr.m.c.9	Binary

For a description of each language object refer to T_UNSIGNED_CPT_BMX IODDT *(see page 180)*.

Free Large Counter Mode Debugging

At a Glance

The table below presents the free large counter mode debugging elements:

Label	Language object	Type
Counter value	%IDr.m.c.2	Digital
Counter valid	%IWr.m.c.0.3	Binary
Counter low	%IWr.m.c.1.0	Binary
Counter in window	%IWr.m.c.1.1	Binary
Counter high	%IWr.m.c.1.2	Binary
Counter in low limit	%IWr.m.c.0.5	Binary
Counter in high limit	%IWr.m.c.0.4	Binary
Capture 0 value	%IDr.m.c.4	Digital
Capture 0 low	%IWr.m.c.1.3	Binary
Capture 0 in window	%IWr.m.c.1.4	Binary
Capture 0 high	%IWr.m.c.1.5	Binary
Capture 0 enable	%QWr.m.c.0.3	Binary
Capture 1 value	%IDr.m.c.16	Digital
Capture 1 low	%IWr.m.c.1.6	Binary
Capture 1 in window	%IWr.m.c.1.7	Binary
Capture 1 high	%IWr.m.c.1.8	Binary
Capture 1 enable	%QWr.m.c.0.4	Binary
Input A state	%Ir.m.c.4	Binary
Input B state	%Ir.m.c.5	Binary
IN_SYNC input	%Ir.m.c.6	Binary
Modulo state	%IWr.m.c.0.1	Binary
Modulo reset	%Qr.m.c.9	Binary
SYNC state	%IWr.m.c.0.2	Binary
SYNC reset	%Qr.m.c.8	Binary
Input EN	%Ir.m.c.7	Binary
EN enable	%QWr.m.c.0.2	Binary
Counter enable	%Qr.m.c.6	Binary
Input REF	%Ir.m.c.8	Binary
REF enable	%QWr.m.c.0.1	Binary
REF force	%QWr.m.c.5	Binary

Label	Language object	Type
Input CAP	%Ir.m.c.9	Binary
Output 0 state	%Ir.m.c.0	Binary
Output 0 cmd	%Qr.m.c.0	Binary
Output 1 state	%Ir.m.c.1	Binary
Output 1 cmd	%Qr.m.c.1	Binary
Counter reset	%Qr.m.c.7	Binary
Output latch 0 state	%Ir.m.c.2	Binary
Output latch 0 enable	%Qr.m.c.2	Binary
Output latch 1 state	%Ir.m.c.3	Binary
Output latch 1 enable	%Qr.m.c.3	Binary
Low threshold value	%QDr.m.c.2	Digital
High threshold value	%QDr.m.c.4	Digital
Compare enable	%QWr.m.c.0.5	Binary
Compare suspend	%QWr.m.c.0.6	Binary

For a description of each language object refer to T_SIGNED_CPT_BMX IODDT *(see page 180)*.

Pulse Width Modulation Mode Debugging

At a Glance

The table below presents the pulse width modulation mode debugging elements:

Label	Language object	Type
Frequency valid	%IWr.m.c.0.3	Binary
Frequency in low limit	%IWr.m.c.0.5	Binary
Frequency in high limit	%IWr.m.c.0.4	Binary
PWM frequency	%QDr.m.c.6	Digital
PWM duty	%QWr.m.c.8	Digital
Input SYNC state	%Ir.m.c.6	Binary
SYNC enable	%QWr.m.c.0.0	Binary
SYNC force	%Qr.m.c.4	Binary
Input EN	%Ir.m.c.7	Binary
EN enable	%QWr.m.c.0.2	Binary
Counter enable	%Qr.m.c.6	Binary
Output latch 0 enable	%Qr.m.c.2	Binary
Output 0 state	%Ir.m.c.0	Binary
Output 0 cmd	%Qr.m.c.0	Binary
Output 1 state	%Ir.m.c.1	Binary
Output 1 cmd	%Qr.m.c.1	Binary

For a description of each language object refer to T_UNSIGNED_CPT_BMX IODDT *(see page 180)*.

Display of BMX EHC xxxx Counting Module Error

12

Subject of this Chapter

This chapter deals with the display of possible errors for the BMX EHC•••• modules.

What Is in This Chapter?

This chapter contains the following topics:

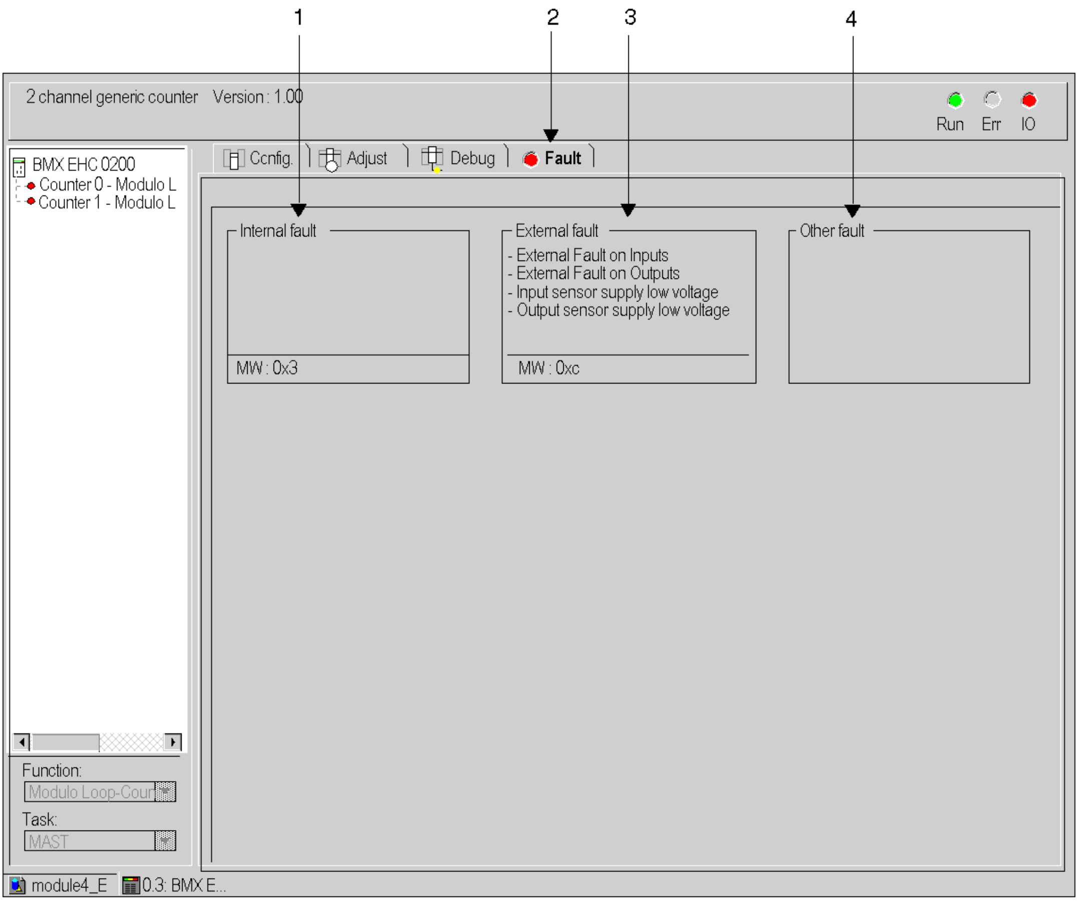
Fault Display Screen for BMX EHC 0200 Counting Modules

At a Glance

This section presents the fault display screen for BMX EHC 0200 counting modules. A module's fault display screen may only be accessed in online mode.

Illustration

The figure below presents the fault display screen for the BMX EHC 0200 module in modulo loop counter mode.



Description of the Screen

The following table presents the various parts of the above screen.

Number	Element	Function
1	Internal faults field	This field displays the module's active internal faults.
2	Tab	The tab in the foreground indicates the current mode. The current mode is therefore the fault display mode in this example.
3	External faults field	This field displays the module's active external faults.
4	Other faults field	This field displays the module's active faults, other than internal and external faults.

Faults Diagnostics Display

At a Glance

The diagnostic screens *(see page 105)* on the module or channel are only accessible in connected mode. When an un-masked fault appears, it is reported:

- in the configuration screen of the rack, with the presence of a red square in the position of the faulty counting module,
- in all screens at module level (**Description** and **Fault** tabs),
  - in the module field with the LED
- in all channel level screens (**Configuration**, **Adjustment**, **Debug** and **Fault** tabs),
  - in the module zone with the LED
  - in the channel zone with the fault LED
- in the fault screen that is accessed by the **Fault** where the fault diagnostics are described.

The fault is also signaled:

- On the module, on the central display,
- by dedicated language objects: CH_ERROR (`%Ir.m.c.ERR`) and MOD_ERROR (`%Ir.m.MOD.ERR`), `%MWr.m.MOD.2`, etc. and status words.

NOTE: Even if the fault is masked, it is reported by the flashing of the **I/O** LED and in the fault screen.

List of Error

At a Glance

The messages displayed on the diagnostics screens are used to assist with debugging. These messages must be concise and are sometimes ambiguous (as different faults may have the same consequences).

These diagnostics are on two levels: module and channel, the latter being the most explicit.

The lists below show the message headings with suggestions for identifying issues.

List of the Module Error Messages

The table below provides a list of the module error messages.

Fault indicated	Possible interpretation and/or action.
Module failure	The module has a error. Check the module mounting. Change the module.
Faulty channel(s)	One or more channels have a fault. Refer to channel diagnostics.
Self-test	The module is running a self-test. Wait until the self-test is complete.
Different hardware and software configurations	There is a lack of compatibility between the module configured and the module in the rack. Make the hardware configuration and the software configuration compatible.
Module is missing or off	Install the module. Fasten the mounting screws.

BMX EHC 0200 Module Error

The table below provides a list of error that may appear on the BMX EHC 0200 module.

Language object	Description
%MWr.m.c.2.0	External fault at inputs
%MWr.m.c.2.1	External fault at outputs
%MWr.m.c.2.4	Internal error or self-testing.
%MWr.m.c.2.5	Configuration Fault
%MWr.m.c.2.6	Communication Error
%MWr.m.c.2.7	Application fault
%MWr.m.c.3.2	Sensor power supply fault
%MWr.m.c.3.3	Actuator supply fault

Language object	Description
%MWr.m.c.3.4	Short circuit on output 0
%MWr.m.c.3.5	Short circuit on output 1

List of Channel Error Messages

The table below gives the list of error messages at channel level.

Fault indicated. Other consequences.	Possible interpretation and/or action.
External fault or counting input fault: • encoder or proximity sensor supply fault • line break or short circuit of at least one encoder differential signal (1A, 1B, 1Z) • specific fault on absolute encoder Outputs are set to 0 in automatic mode. **Invalid measurement** message.	Check the sensor connections. Check the sensor power supply. Check the sensor operation. Delete the fault and acknowledge if the fault storing is configured. Counting pulses or incremental encoder: preset or reset to acknowledge the **Invalid measurement** message.
Counting application fault: • measurement overrun • overspeed Outputs are set to 0 in automatic mode. **Invalid measurement** message.	Diagnose the fault more precisely (external causes). Check the application again, if necessary. Delete the fault and acknowledge if the fault storing is configured. Counting pulses or incremental encoder: preset or reset to 0 to acknowledge the **Invalid measurement** message.
Auxiliary input/output fault: • power supply • short circuit of at least one output Outputs are set to 0 in automatic mode	Check the output connections Check the input/output power supply (24V) Diagnose the fault more precisely (external causes) Delete the fault and acknowledge if the fault storing is configured
Internal error or channel self-testing: • module faulty • module missing or off • module running self-test	Module fault has gone down to channel level. Refer to module level diagnostics.
Different hardware and software configurations	Module fault has gone down to channel level. Refer to module level diagnostics.

Fault indicated. Other consequences.	**Possible interpretation and/or action.**
Invalid software configuration: ● incorrect constant ● bit combination not associated with any configuration	Check and modify the configuration constants.
Communication error	Check the connections between the racks.
Application fault: refusal to configure or adjust	Diagnose the fault more precisely.

The Language Objects of the Counting Function

13

Subject of this Chapter

This chapter describes the language objects associated to the counting tasks as well as the different ways of using them.

What Is in This Chapter?

This chapter contains the following sections:

13.1 The Language Objects and IODDT of the Counting Function

Subject of this Section

This section describes the general features of the language objects and IODDT of the counting function.

What Is in This Section?

This section contains the following topics:

Introducing Language Objects for Application-Specific Counting

General

The counting modules have only two associated IODDTs. These IODDTs are predefined by the manufacturer and contains language objects for inputs/outputs belonging to the channel of an application-specific module.

The IODDT associated with the counting modules are of T_ Unsigned_CPT_BMX and T_Signed_CPT_BMX types.

NOTE: IODDT variables can be created in two different ways:

- Using the **I/O objects** *(see Unity Pro, Operating Modes)* tab.
- Using the Data Editor *(see Unity Pro, Operating Modes)*.

Language Object Types

Each IODDT contains a set of language objects allowing its operation to be controlled and checked.

There are two types of language objects:

- **Implicit Exchange Objects:** these objects are automatically exchanged on each cycle revolution of the task associated with the module.
- **Explicit Exchange Objects:** these objects are exchanged on the application's request, using explicit exchange instructions.

Implicit exchanges concern the inputs/outputs of the module (measurement results, information and commands). These exchanges enable the debugging of the counting modules.

Explicit exchanges enable the module to be set and diagnosed.

Implicit Exchange Language Objects Associated with the Application-Specific Function

At a Glance

An integrated application-specific interface or the addition of a module automatically enhances the language objects application used to program this interface or module.

These objects correspond to the input/output images and software data of the module or integrated application-specific interface.

Reminders

The module inputs (%I and %IW) are updated in the PLC memory at the start of the task, the PLC being in RUN or STOP mode.

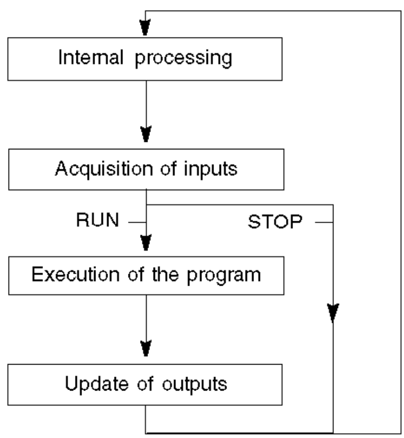
The outputs (%Q and %QW) are updated at the end of the task, only when the PLC is in RUN mode.

NOTE: When the task occurs in STOP mode, either of the following are possible, depending on the configuration selected:

- outputs are set to fallback position (fallback mode)
- outputs are maintained at their last value (maintain mode)

Figure

The following diagram shows the operating cycle of a PLC task (cyclical execution).



Explicit Exchange Language Objects Associated with the Application-Specific Function

Introduction

Explicit exchanges are performed at the user program's request using these instructions:

- READ_STS *(see Unity Pro, I/O Management, Block Library)* (read status words)
- WRITE_CMD *(see Unity Pro, I/O Management, Block Library)* (write command words)
- WRITE_PARAM *(see Unity Pro, I/O Management, Block Library)* (write adjustment parameters)
- READ_PARAM *(see Unity Pro, I/O Management, Block Library)* (read adjustment parameters)
- SAVE_PARAM *(see Unity Pro, I/O Management, Block Library)* (save adjustment parameters)
- RESTORE_PARAM *(see Unity Pro, I/O Management, Block Library)* (restore adjustment parameters)

These exchanges apply to a set of %MW objects of the same type (status, commands or parameters) that belong to a channel.

These objects can:

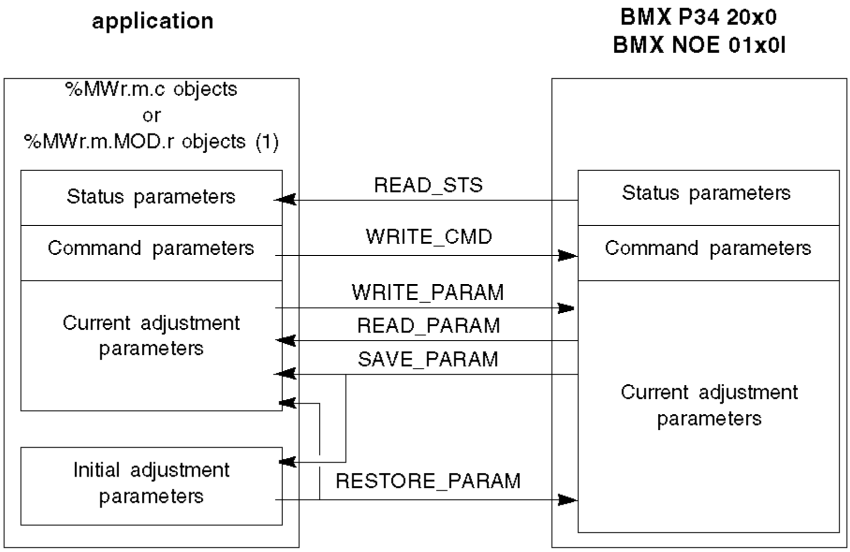
- provide information about the module (for example, type of error detected in a channel)
- have command control of the module (for example, switch command)
- define the module's operating modes (save and restore adjustment parameters in the process of application)

NOTE: To avoid several simultaneous explicit exchanges for the same channel, it is necessary to test the value of the word EXCH_STS (`%MWr.m.c.0`) of the IODDT associated to the channel before calling any EF addressing this channel.

NOTE: Explicit Exchanges are not supported when Modicon M340 Analog and Digital I/O modules are configured behind a M340 Ethernet Remote I/O adapter module in a Quantum EIO Ethernet Configuration. As a consequence, it is not possible to setup a module's parameters from the PLC application during operation.

General Principle for Using Explicit Instructions

The diagram below shows the different types of explicit exchanges that can be made between the application and module.



(1) Only with READ_STS and WRITE_CMD instructions.

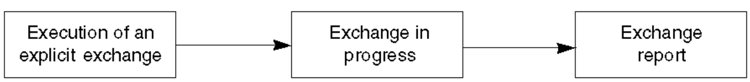
Managing Exchanges

During an explicit exchange, check performance to see that the data is only taken into account when the exchange has been correctly executed.

To do this, two types of information is available:

- information concerning the exchange in progress *(see page 177)*
- the exchange report *(see page 178)*

The following diagram describes the management principle for an exchange.



NOTE: In order to avoid several simultaneous explicit exchanges for the same channel, it is necessary to test the value of the word EXCH_STS (`%MWr.m.c.0`) of the IODDT associated to the channel before calling any EF addressing this channel.

Management of Exchanges and Reports with Explicit Objects

At a Glance

When data is exchanged between the PLC memory and the module, the module may require several task cycles to acknowledge this information. All IODDTs use two words to manage exchanges:

- `EXCH_STS` (`%MWr.m.c.0`): exchange in progress
- `EXCH_RPT` (`%MWr.m.c.1`): report

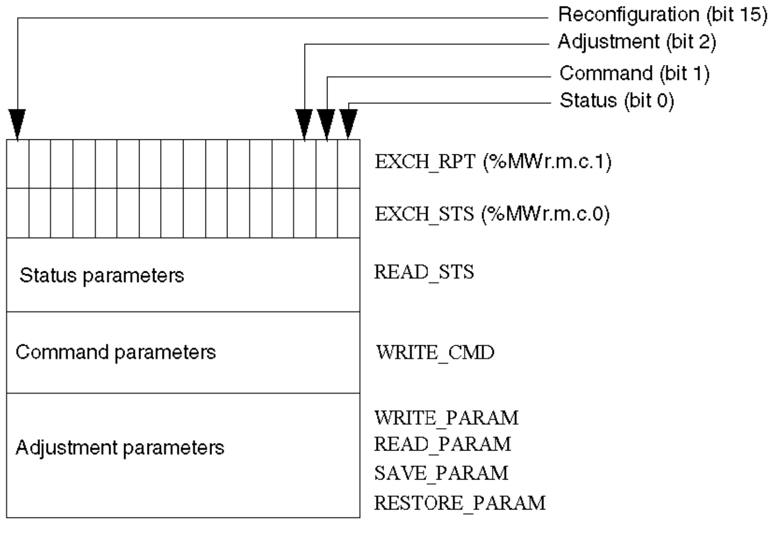
NOTE:

Depending on the localization of the module, the management of the explicit exchanges (`%MW0.0.MOD.0.0` for example) will not be detected by the application:

- For in-rack modules, explicit exchanges are done immediately on the local PLC Bus and are finished before the end of the execution task. So, the `READ_STS`, for example, is always finished when the `%MW0.0.mod.0.0` bit is checked by the application.
- For remote bus (Fipio for example), explicit exchanges are not synchronous with the execution task, so the detection is possible by the application.

Illustration

The illustration below shows the different significant bits for managing exchanges:



Description of Significant Bits

Each bit of the words EXCH_STS (%MWr.m.c.0) and EXCH_RPT (%MWr.m.c.1) is associated with a type of parameter:

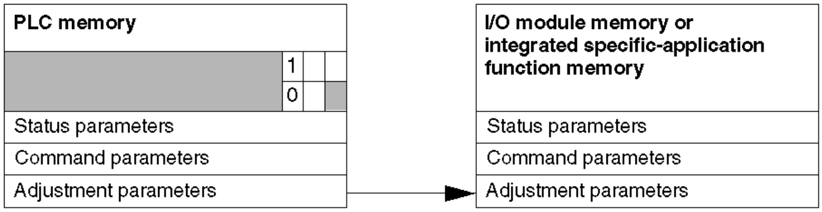
- Rank 0 bits are associated with the status parameters:
  - The STS_IN_PROGR bit (%MWr.m.c.0.0) indicates whether a read request for the status words is in progress.
  - The STS_ERR bit (%MWr.m.c.1.0) specifies whether a read request for the status words is accepted by the module channel.
- Rank 1 bits are associated with the command parameters:
  - The CMD_IN_PROGR bit (%MWr.m.c.0.1) indicates whether command parameters are being sent to the module channel.
  - The CMD_ERR bit (%MWr.m.c.1.1) specifies whether the command parameters are accepted by the module channel.
- Rank 2 bits are associated with the adjustment parameters:
  - The ADJ_IN_PROGR bit (%MWr.m.c.0.2) indicates whether the adjustment parameters are being exchanged with the module channel (via WRITE_PARAM, READ_PARAM, SAVE_PARAM, RESTORE_PARAM).
  - The ADJ_ERR bit (%MWr.m.c.1.2) specifies whether the adjustment parameters are accepted by the module. If the exchange is correctly executed, the bit is set to 0.
- Rank 15 bits indicate a reconfiguration on channel **c** of the module from the console (modification of the configuration parameters + cold start-up of the channel).
- The *r*, *m* and *c* bits indicates the following elements:
  - the **r** bit represents the rack number.
  - The **m** bit represents the position of the module in the rack.
  - The **c** bit represents the channel number in the module.

NOTE: **r** represents the rack number, **m** the position of the module in the rack, while **c** represents the channel number in the module.

NOTE: Exchange and report words also exist at module level EXCH_STS (%MWr.m.MOD) and EXCH_RPT (%MWr.m.MOD.1) as per IODDT type T_GEN_MOD.

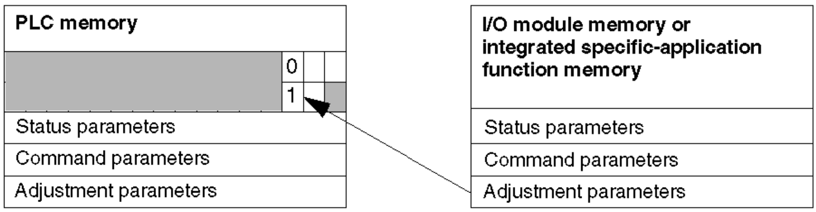
Example

Phase 1: Sending data by using the `WRITE_PARAM` instruction



When the instruction is scanned by the PLC processor, the **Exchange in progress** bit is set to 1 in `%MWr.m.c`.

Phase 2: Analysis of the data by the I/O module and report.



When the data is exchanged between the PLC memory and the module, acknowledgement by the module is managed by the `ADJ_ERR` bit (`%MWr.m.c.1.2`).

This bit makes the following reports:
- **0:** correct exchange
- **1:** faulty exchange)

NOTE: There is no adjustment parameter at module level.

Execution Indicators for an Explicit Exchange: EXCH_STS

The table below shows the control bits of the explicit exchanges: `EXCH_STS` (`%MWr.m.c.0`)

Standard symbol	Type	Access	Meaning	Address
STS_IN_PROGR	BOOL	R	Reading of channel status words in progress	%MWr.m.c.0.0
CMD_IN_PROGR	BOOL	R	Command parameters exchange in progress	%MWr.m.c.0.1
ADJ_IN_PROGR	BOOL	R	Adjust parameters exchange in progress	%MWr.m.c.0.2

Standard symbol	Type	Access	Meaning	Address
RECONF_IN_PROGR	BOOL	R	Reconfiguration of the module in progress	%MWr.m.c.0.15

NOTE: If the module is not present or is disconnected, explicit exchange objects (`READ_STS` for example) are not sent to the module (`STS_IN_PROG` (%MWr.m.c.0.0) = 0), but the words are refreshed.

Explicit Exchange Report: EXCH_RPT

The table below shows the report bits: `EXCH_RPT` (`%MWr.m.c.1`)

Standard symbol	Type	Access	Meaning	Address
STS_ERR	BOOL	R	Error reading channel status words (1 = failure)	%MWr.m.c.1.0
CMD_ERR	BOOL	R	Error during a command parameter exchange (1 = failure)	%MWr.m.c.1.1
ADJ_ERR	BOOL	R	Error during an adjust parameter exchange (1 = failure)	%MWr.m.c.1.2
RECONF_ERR	BOOL	R	Error during reconfiguration of the channel (1 = failure)	%MWr.m.c.1.15

Counting Module Use

The following table describes the steps realised between a Couting Module and the system after a power-on.

Step	Action
1	Power on.
2	The system sends the configuration parameters.
3	The system sends the adjust parameters by WRITE_PARAM method. **Note:** When the operation is finished, the bit %MWr.m.c.0.2 switches to 0.

If, in the begining of your application, you use a WRITE_PARAM command, you must wait until the bit %MWr.m.c.0.2 switches to 0.

13.2 Language Objects and IODDT Associated with the Counting Function of the BMX EHC xxxx Modules.

Subject of this Section

This section presents the language objects and IODDTs associated with the counting function of BMX EHC •••• modules.

What Is in This Section?

This section contains the following topics:

Details of Implicit Exchange Objects for the T_Unsigned_CPT_BMX and T_Signed_CPT_BMX-types IODDTs

At a Glance

The tables below present the T_Unsigned_CPT_BMX and T_Signed_CPT_BMX-types IODDTs implicit exchange objects which are applicable to all **BMX EHC ••••** counting modules.

Counter Value and Sensor Values

The table below presents the various IODDT implicit exchange objects:

Standard symbol	Type	Access	Meaning	Language object
COUNTER_CURRENT_VALUE	DINT	R	Current counter value	%IDr.m.c.2
CAPT_0_VALUE	DINT	R	Counter value when captured in register 0	%IDr.m.c.4
CAPT_1_VALUE	DINT	R	Counter value when captured in register 1	%IDr.m.c.6
COUNTER_VALUE	DINT	R	Current counter value during event	%IDr.m.c.12
CAPT_0_VAL	DINT	R	Capture value 0	%IDr.m.c.14
CAPT_1_VAL	DINT	R	Capture value 1	%IDr.m.c.16

%Ir.m.c.d Word

The table below presents the meanings of the %Ir.m.c.d words:

Standard symbol	Type	Access	Meaning	Language object
CH_ERROR	BOOL	R	Channel error	%Ir.m.c.ERR
OUTPUT_0_Echo	BOOL	R	Logical state of output 0	%Ir.m.c.0
OUTPUT_1_Echo	BOOL	R	Logical state of output 1	%Ir.m.c.1
OUTPUT_BLOCK_0	BOOL	R	State of output block 0	%Ir.m.c.2
OUTPUT_BLOCK_1	BOOL	R	State of output block 1	%Ir.m.c.3
INPUT_A	BOOL	R	Physical state of IN_A input	%Ir.m.c.4
INPUT_B	BOOL	R	Physical state of IN_B input	%Ir.m.c.5
INPUT_SYNC	BOOL	R	Physical state of the IN_SYNC input (or IN_AUX)	%Ir.m.c.6
INPUT_EN	BOOL	R	Physical state of IN_EN input (enable)	%Ir.m.c.7
INPUT_REF	BOOL	R	Physical state of the IN_REF input (preset)	%Ir.m.c.8
INPUT_CAPT	BOOL	R	Physical state of IN_CAP input (capture)	%Ir.m.c.9

Counter Status, %IWr.m.c.0 Word

The following table presents the meanings of the bits of the %IWr.m.c.0 status word:

Standard symbol	Type	Access	Meaning	Language object
RUN	BOOL	R	The counter operates in counting mode only	%IWr.m.c.0.0
MODULO_FLAG	BOOL	R	Flag set to 1 by a modulo switch event	%IWr.m.c.0.1
SYNC_REF_FLAG	BOOL	R	Flag set to 1 by a preset or synchronization event	%IWr.m.c.0.2
VALIDITY	BOOL	R	The current numerical value is valid	%IWr.m.c.0.3
HIGH_LIMIT	BOOL	R	The current numerical value is locked at the upper threshold value	%IWr.m.c.0.4
LOW_LIMIT	BOOL	R	The current numerical value is locked at the lower threshold value	%IWr.m.c.0.5

Comparison Status, %IWr.m.c.1 Word

The following table presents the meanings of the bits of the %IWr.m.c.1 status word:

Standard symbol	Type	Access	Meaning	Language object
COUNTER_LOW	BOOL	R	Current counter value less than lower threshold (%QDr.m.c.2)	%IWr.m.c.1.0
COUNTER_WIN	BOOL	R	Current counter value is between lower threshold (%QDr.m.c.2) and upper threshold (%QDr.m.c.4)	%IWr.m.c.1.1
COUNTER_HIGH	BOOL	R	Current counter value greater than upper threshold (%QDr.m.c.4)	%IWr.m.c.1.2
CAPT_0_LOW	BOOL	R	Value captured in register 0 is less than lower threshold (%QDr.m.c.2)	%IWr.m.c.1.3
CAPT_0_WIN	BOOL	R	Value captured in register 0 is between lower threshold (%QDr.m.c.2) and upper threshold (%QDr.m.c.4)	%IWr.m.c.1.4
CAPT_0_HIGH	BOOL	R	Value captured in register 0 is greater than upper threshold (%QDr.m.c.4)	%IWr.m.c.1.5
CAPT_1_LOW	BOOL	R	Value captured in register 1 is less than lower threshold (%QDr.m.c.2)	%IWr.m.c.1.6
CAPT_1_WIN	BOOL	R	Value captured in register 1 is between lower threshold (%QDr.m.c.2) and upper threshold (%QDr.m.c.4)	%IWr.m.c.1.7
CAPT_1_HIGH	BOOL	R	Value captured in register 1 is greater than upper threshold (%QDr.m.c.4)	%IWr.m.c.1.8

Event Sources, %IWr.m.c.10 Word

The following table presents the meanings of the bits of the %IWr.m.c.10 word:

Standard symbol	Type	Access	Meaning	Language object
EVT_SOURCES	INT	R	Event sources field	%IWr.m.c.10
EVT_RUN	BOOL	R	Event due to start of counter.	%IWr.m.c.10.0
EVT_MODULO	BOOL	R	Event due to modulo switch	%IWr.m.c.10.1
EVT_SYNC_PRESET	BOOL	R	Event due to synchronization or preset	%IWr.m.c.10.2
EVT_COUNTER_LOW	BOOL	R	Event due to counter value being less than lower threshold	%IWr.m.c.10.3
EVT_COUNTER_WINDOW	BOOL	R	Event due to counter value being between the two thresholds	%IWr.m.c.10.4
EVT_COUNTER_HIGH	BOOL	R	Event due to counter value being greater than upper threshold	%IWr.m.c.10.5
EVT_CAPT_0	BOOL	R	Event due to capture function 0	%IWr.m.c.10.6
EVT_CAPT_1	BOOL	R	Event due to capture function 1	%IWr.m.c.10.7
EVT_OVERRUN	BOOL	R	Warning: lost event(s)	%IWr.m.c.10.8

Output Thresholds and Frequency

The table below presents the various IODDT implicit exchange objects:

Standard symbol	Type	Access	Meaning	Language object
LOWER_TH_VALUE	DINT	R/W	Lower threshold value	%QDr.m.c.2
UPPER_TH_VALUE	DINT	R/W	Upper threshold value	%QDr.m.c.4
PWM_FREQUENCY	DINT	R/W	Output frequency value (unit = 0.1 Hz)	%QDr.m.c.6
PWM_DUTY	INT	R/W	Duty cycle value of the output frequency (unit = 5%)	%QDr.m.c.8

%Qr.m.c.d Words

The following table presents the meanings of the bits of the %Qr.m.c.d words:

Standard symbol	Type	Access	Meaning	Language object
OUTPUT_0	BOOL	R/W	Forces OUTPUT_0 to level 1	%Qr.m.c.0
OUTPUT_1	BOOL	R/W	Forces OUTPUT_1 to level 1	%Qr.m.c.1
OUTPUT_BLOCK_0_ENABLE	BOOL	R/W	Implementation of output 0 function block	%Qr.m.c.2
OUTPUT_BLOCK_1_ENABLE	BOOL	R/W	Implementation of output 1 function block	%Qr.m.c.3

Standard symbol	Type	Access	Meaning	Language object
FORCE_SYNC	BOOL	R/W	Counting function synchronization and start	%Qr.m.c.4
FORCE_REF	BOOL	R/W	Set to preset counter value	%Qr.m.c.5
FORCE_ENABLE	BOOL	R/W	Implementation of counter	%Qr.m.c.6
FORCE_RESET	BOOL	R/W	Reset counter	%Qr.m.c.7
SYNC_RESET	BOOL	R/W	Reset SYNC_REF_FLAG	%Qr.m.c.8
MODULO_RESET	BOOL	R/W	Reset MODULO_FLAG	%Qr.m.c.9

FUNCTIONS_ENABLING, %QWr.m.c.0 Word

The following table presents the meanings of the bits of the %QWr.m.c.0 words:

Standard symbol	Type	Access	Meaning	Language object
VALID_SYNC	BOOL	R/W	Synchronization and start authorization for the counting function via the IN_SYNC input	%QWr.m.c.0.0
VALID_REF	BOOL	R/W	Operation authorization for the internal preset function	%QWr.m.c.0.1
VALID_ENABLE	BOOL	R/W	Authorization of the counter enable via the IN_EN input	%QWr.m.c.0.2
VALID_CAPT_0	BOOL	R/W	Capture authorization in the capture0 register	%QWr.m.c.0.3
VALID_CAPT_1	BOOL	R/W	Capture authorization in the capture1 register	%QWr.m.c.0.4
COMPARE_ENABLE	BOOL	R/W	Comparators operation authorization	%QWr.m.c.0.5
COMPARE_SUSPEND	BOOL	R/W	Comparator frozen at its last value	%QWr.m.c.0.6

EVENT_SOURCES_ENABLING, %QWr.m.c.1 Word

The following table presents the meanings of the bits of the %QWr.m.c.1 words:

Standard symbol	Type	Access	Meaning	Language object
EVT_RUN_ENABLE	BOOL	R/W	EVENT task call at start of the counting function	%QWr.m.c.1.0
EVT_MODULO_ENABLE	BOOL	R/W	EVENT task call when there is a counter reversal	%QWr.m.c.1.1
EVT_REF_ENABLE	BOOL	R/W	EVENT task call during counter synchronization or preset	%QWr.m.c.1.2
EVT_COUNTER_LOW_ENABLE	BOOL	R/W	EVENT task call when the counter value is less than lower threshold	%QWr.m.c.1.3
EVT_COUNTER_WINDOW_ENABLE	BOOL	R/W	EVENT task call when the counter is between the lower and upper threshold	%QWr.m.c.1.4
EVT_COUNTER_HIGH_ENABLE	BOOL	R/W	EVENT task call when the counter value is greater than the upper threshold	%QWr.m.c.1.5
EVT_CAPT_0_ENABLE	BOOL	R/W	EVENT task call during capture in register 0	%QWr.m.c.1.6
EVT_CAPT_1_ENABLE	BOOL	R/W	EVENT task call during capture in register 1	%QWr.m.c.1.7

Details of the Explicit Exchange Objects for the T_CPT_BMX-type IODDT

At a Glance

This section presents the explicit exchange objects for the T_Unsigned_CPT_BMX and T_Signed_CPT_BMX- types IODDTs which are applicable to all BMX EHC •••• counting modules. They includes word type objects whose bits have a specific meaning. These objects are described in detail below.

Sample variable declaration: T_Unsigned_CPT_BMX and T_Signed_CPT_BMX-types IODDT_VAR1.

NOTE:

- in general, the meaning of the bits is given for bit status 1.
- not all bits are used.

Preset Values

The table below shows the meaning of the status bits.

Standard symbol	Type	Access	Meaning	Language object
MODULO_VALUE	DINT	R/W	Modulo value	%MDr.m.c.4
PRESET_VALUE	DINT	R/W	Preset value	%MDr.m.c.6
CALIBRATION_FACTOR	INT	R/W	Calibration factor -10% to +10% (unit = 0.1%)	%MWr.m.c.8
SLACK_VAL	INT	R/W	Offset value	%MWr.m.c.9

Exchange Status: EXCH_STS

The table below shows the meaning of channel exchange status bits from the EXCH_STS channel (%MWr.m.c.0).

Standard symbol	Type	Access	Meaning	Language object
STS_IN_PROG	BOOL	R	Status parameter read in progress	%MWr.m.c.0.0
ADJ_IN_PROG	BOOL	R	Adjust parameter exchange in progress	%Mwr.m.c.0.2
RECONF_IN_PROG	BOOL	R	Reconfiguration in progress	%MWr.m.c.0.15

Channel Report: EXCH_RPT

The following table presents the meanings of the report bits of the EXCH_RPT channel (%MWr.m.c.1).

Standard symbol	Type	Access	Meaning	Language object
STS_ERR	BOOL	R	Error while reading channel status	%MWr.m.c.1.0
ADJ_ERR	BOOL	R	Error while adjusting the channel	%Mwr.m.c.1.2
RECONF_ERR	BOOL	R	Error while reconfiguring the channel	%MWr.m.c.1.15

Channel Error: CH_FLT

The table below presents the meaning of the error bits on the CH_FLT channel (%MWr.m.c.2).

Standard symbol	Type	Access	Meaning	Language object
EXTERNAL_FLT_INPUTS	BOOL	R	External error at inputs	%MWr.m.c.2.0
EXTERNAL_FLT_OUTPUTS	BOOL	R	External error at outputs	%MWr.m.c.2.1
INTERNAL_FLT	BOOL	R	Internal error: channel inoperative	%MWr.m.c.2.4
CONF_FLT	BOOL	R	Hardware or software configuration error	%MWr.m.c.2.5
COM_FLT	BOOL	R	Bus Communication error	%MWr.m.c.2.6
APPLI_FLT	BOOL	R	Application error	%MWr.m.c.2.7

Channel Error: %MWr.m.c.3

The table below presents the meaning of the error bits on the %MWr.m.c.3 word.

Standard symbol	Type	Access	Meaning	Language object
SENSOR_SUPPLY	BOOL	R	Low input power supply for the sensors	%MWr.m.c.3.2
ACTUATOR_SUPPLY_FLT	BOOL	R	Output power supply failure	%MWr.m.c.3.3
SHORT_CIRCUIT_OUT_0	BOOL	R	Short circuit on output 0	%MWr.m.c.3.4
SHORT_CIRCUIT_OUT_1	BOOL	R	Short circuit on output 1	%MWr.m.c.3.5

13.3 Device DDTs Associated with the Counting Function of the BMX EHC xxxx Modules.

Counter Device DDT Names

Introduction

This topic describes the Unity Pro **Counter Device DDT**.

The default device DDT name contains the following information:
- module input and or output (***X*** symbol)
- module insertion number (***#*** symbol)

Example: `MOD_CPT_`***x_#***

The default device DDT type contains the following information:
- platform with:
  - M for Modicon M340
- device type (CPT for counter)
- function (STD for standard)
- direction:
  - IN
  - OUT
- max channel (2 or 8)

Example: For a Modicon M340 with 2 standard inputs: T_M_CPT_STD_IN_2

Adjustment Parameter limitation

Adjustment parameters cannot be changed from the PLC application during operation (no support of `READ_PARAM`, `WRITE_PARAM`, `SAVE_PARAM`, `RESTORE_PARAM`).

Modifying the adjustment parameters of a channel from Unity Pro during a CCOTF operation causes the channel to be re-initialized.

The concerned parameters are:
- `PRESET_VALUE`
  Preset value
- `CALIBRATION_FACTOR`
  Calibration Factor
- `MODULO_VALUE`
  Modulo value
- `SLACK_VAL`
  Offset value
- `HYSTERESIS_VALUE`
  Hysteresis value

List of Implicit Device DDT

The following table shows the list of the Modicon M340 devices and their corresponding device DDT name and type:

Device DDT Name	Device DDT Type	Modicon M340 Devices
MOD_CPT_2_#	`T_M_CPT_STD_IN_2`	BMX EHC 0200
MOD_CPT_8_#	`T_M_CPT_STD_IN_8`	BMX EHC 0800

Implicit Device DDT Description

The following table shows the `T_M_CPT_STD_IN_x` status word bits:

Standard Symbol	Type	Meaning	Access
MOD_HEALTH	BOOL	0 = the module has a detected error	read
		1 = the module is operating correctly	
MOD_FLT	BYTE	internal detected errors byte of the module	read
CPT_CH_IN	ARRAY [0...x-1] of `T_M_CPT_STD_CH_IN`	Array of structure	

The following table shows the `T_M_CPT_STD_CH_IN_x`[0...x-1] status word bits:

Standard Symbol	Type	Bit	Meaning	Access
FCT_TYPE	WORD		1 = Frequency	read
			2 = EvtCounting	
			3 = PeriodMeasuring	
			4 = Ratio1	
			5 = Ratio2	
			6 = OneShotCounter	
			7 = ModuleLoopCounter	
			8 = FreeLargeCounter	
			9 = PulseWidthModulation	
			10 = UpDownCounting	
			11 = DualPhaseCounting	
CH_HEALTH	BOOL		0 = channel is inactive	read
			1 = channel is active	
ST_OUTPUT_0_ECHO	EBOOL		logical state of output 0	read

Standard Symbol		Type	Bit	Meaning	Access
ST_OUTPUT_1_ECHO		EBOOL		logical state of output 1	read
ST_OUTPUT_BLOCK_0		EBOOL		status of physical counting output block 0	read
ST_OUTPUT_BLOCK_1		EBOOL		status of physical counting output block 1	read
ST_INPUT_A		EBOOL		status of physical counting input A	read
ST_INPUT_B		EBOOL		status of physical counting input B	read
ST_INPUT_SYNC		EBOOL		physical state of the IN_SYNC input (or IN_AUX)	read
ST_INPUT_EN		EBOOL		physical state of IN_EN input (enable)	read
ST_INPUT_REF		EBOOL		physical state of the IN_REF input (preset)	read
ST_INPUT_CAPT		EBOOL		physical state of IN_CAP input (capture)	read
COUNTER_STATUS [INT]	RUN	BOOL	0	the counter operates in counting mode only	read
	MODULO_FLAG	BOOL	1	flag set to 1 by a modulo switch event	read
	SYNC_REF_FLAG	BOOL	2	flag set to 1 by a preset or synchronization event	read
	VALIDITY	BOOL	3	the current numerical value is valid	read
	HIGH_LIMIT	BOOL	4	the current numerical value is locked at the upper threshold value	read
	LOW_LIMIT	BOOL	5	the current numerical value is locked at the lower threshold value	read

Standard Symbol		Type	Bit	Meaning	Access
COMPARE_STATUS [INT]	COUNTER_LOW	BOOL	0	current counter value less than lower threshold (LOWER_TH_VALUE)	read
	COUNTER_WIN	BOOL	1	current counter value is between lower threshold (LOWER_TH_VALUE) and upper threshold (UPPER_TH_VALUE)	read
	COUNTER_HIGH	BOOL	2	current counter value greater than upper threshold (UPPER_TH_VALUE)	read
	CAPT_0_LOW	BOOL	3	Value captured in register 0 is less than lower threshold (LOWER_TH_VALUE)	read
	CAPT_0_WIN	BOOL	4	Value captured in register 0 is between lower threshold (LOWER_TH_VALUE) and upper threshold (UPPER_TH_VALUE)	read
	CAPT_0_HIGH	BOOL	5	Value captured in register 0 is greater than upper threshold (UPPER_TH_VALUE)	read
	CAPT_1_LOW	BOOL	6	Value captured in register 1 is less than lower threshold (LOWER_TH_VALUE)	read
	CAPT_1_WIN	BOOL	7	Value captured in register 1 is between lower threshold (LOWER_TH_VALUE) and upper threshold (UPPER_TH_VALUE)	read
	CAPT_1_HIGH	BOOL	8	Value captured in register 1 is greater than upper threshold (UPPER_TH_VALUE)	read
COUNTER_CURRENT_VALUE_S[1]		DINT		Current counter value during event	read

Standard Symbol		Type	Bit	Meaning	Access
CAPT_0_VALUE_S[1]		DINT		Value captured in register 0	read
CAPT_1_VALUE_S[1]		DINT		Value captured in register 1	read
COUNTER_CURRENT_VALUE_US[2]		UDINT		Current counter value during event	read
CAPT_0_VALUE_US[2]		UDINT		Value captured in register 0	read
CAPT_1_VALUE_US[2]		UDINT		Value captured in register 1	read
OUTPUT_0		EBOOL		forces OUTPUT_0 to level 1	read / write
OUTPUT_1		EBOOL		forces OUTPUT_1 to level 1	read / write
OUTPUT_BLOCK_0_ENABLE		EBOOL		implementation of output 0 function block	read / write
OUTPUT_BLOCK_1_ENABLE		EBOOL		implementation of output 1 function block	read / write
FORCE_SYNC		EBOOL		counting function synchronization and start	read / write
FORCE_REF		EBOOL		set to preset counter value	read / write
FORCE_ENABLE		EBOOL		implementation of counter	read / write
FORCE_RESET		EBOOL		reset counter	read / write
SYNC_RESET		EBOOL		reset SYNC_REF_FLAG	read / write
MODULO_RESET		EBOOL		reset MODULO_FLAG	read / write

Standard Symbol		Type	Bit	Meaning	Access
FUNCTIONS_ENABLING [INT]	VALID_SYNC	BOOL	0	synchronization and start authorization for the counting function via the IN_SYNC input	read / write
	VALID_REF	BOOL	1	operation authorization for the internal preset function	read / write
	VALID_ENABLE	BOOL	2	authorization of the counter enable via the IN_EN input	read / write
	VALID_CAPT_0	BOOL	3	capture authorization in the capture 0 register	read / write
	VALID_CAPT_1	BOOL	4	capture authorization in the capture 1 register	read / write
	COMPARE_ENABLE	BOOL	5	comparators operation authorization	read / write
	COMPARE_SUSPEND	BOOL	6	comparator frozen at its last value	read / write
LOWER_TH_VALUE_S[1]		DINT		lower threshold value	read / write
UPPER_TH_VALUE_S[1]		DINT		upper threshold value	read / write
PWM_FREQUENCY_S[1]		DINT		output frequency value (unit = 0.1 Hz)	read / write
LOWER_TH_VALUE_US[2]		UDINT		lower threshold value	read / write
UPPER_TH_VALUE_US[2]		UDINT		upper threshold value	read / write
PWM_FREQUENCY_US[2]		UDINT		output frequency value (unit = 0.1 Hz)	read / write
PWM_DUTY		INT		duty cycle value of the output frequency (unit = 5%)	read / write
1: Signed application specific function (ASF) must be used **2:** Unsigned application specific function (ASF) must be used					

Here below is all the signed ASF that must be used with a counter ••• EHC 0200:

- Free Large counter Mode
- Ratio 1
- Ratio 2

Here below is all the unsigned ASF that must be used with a counter ••• EHC 0200:

- Event Counting Mode
- Frequency Mode
- Modulo Loop Counter Mode

- One Shot Counter Mode
- Period Measuring Mode
- Pulse Width Modulation Mode

Here below is all the signed ASF that must be used with a counter ••• EHC 0800:

- Up Down Counting Mode

Here below is all the unsigned ASF that must be used with a counter ••• EHC 0800:

- Event Counting Mode
- Frequency Mode
- Modulo Loop Counter Mode
- One Shot Counter Mode

Explicit Device DDT instances Description

Explicit exchanges (Read Status) - only applicable to Modicon M340 I/O channels - are managed with `READ_STS_QX` EFB instance.

- Targeted channel address (`ADDR`) can be managed with ADDMX *(see Unity Pro, Communication, Block Library)* EF (connect `ADDMX OUT` to `ADDR`)
- READ_STS_QX *(see Unity Pro, I/O Management, Block Library)* output parameter (`STATUS`) can be connected to a "`T_M_xxx_yyy_CH_STS`" DDT instance (variable to be created manually), where:
  - `xxx` represents the device type
  - `yyy` represents the function

  Example: `T_M_CPT_STD_CH_STS`

The following table shows the `T_M_CPT_STD_CH_STS` status word bits:

Type	Type	Access
STRUCT	`T_M_CPT_STD_CH_STS`	

The following table shows the `T_M_CPT_STD_CH_STS` status word bits:

Standard Symbol		Type	Bit	Meaning	Access
CH_FLT [INT]	EXTERNAL_FLT_INPUTS	BOOL	0	external detected error at inputs	read
	EXTERNAL_FLT_OUTPUTS	BOOL	1	external detected error at outputs	read
	INTERNAL_FLT	BOOL	4	internal detected error: channel inoperative	read
	CONF_FLT	BOOL	5	hardware or software configuration detected error	read
	COM_FLT	BOOL	6	bus communication detected error	read
	APPLI_FLT	BOOL	7	application detected error	read
	COM_EVT_FLT	BOOL	8	communication event detected fault	read
	OVR_EVT_CPU	BOOL	9	CPU overflow event	read
	OVR_CPT_CH	BOOL	10	counter channel overflow	read
CH_FLT_2 [INT]	SENSOR_SUPPLY	BOOL	2	low input power supply for the sensors	read
	ACTUATOR_SUPPLY	BOOL	3	output power supply loss	read
	SHORT_CIRCUIT_OUT_0	BOOL	4	short circuit on output 0	read
	SHORT_CIRCUIT_OUT_1	BOOL	5	short circuit on output 1	read

13.4 The IODDT Type T_GEN_MOD Applicable to All Modules

Details of the Language Objects of the IODDT of Type T_GEN_MOD

Introduction

All the modules of Modicon M340 PLCs have an associated IODDT of type T_GEN_MOD.

Observations

In general, the meaning of the bits is given for bit status 1. In specific cases an explanation is given for each status of the bit.

Some bits are not used.

List of Objects

The table below presents the objects of the IODDT.

Standard Symbol	Type	Access	Meaning	Address
MOD_ERROR	BOOL	R	Module error bit	%Ir.m.MOD.ERR
EXCH_STS	INT	R	Module exchange control word	%MWr.m.MOD.0
STS_IN_PROGR	BOOL	R	Reading of status words of the module in progress	%MWr.m.MOD.0.0
EXCH_RPT	INT	R	Exchange report word	%MWr.m.MOD.1
STS_ERR	BOOL	R	Event when reading module status words	%MWr.m.MOD.1.0
MOD_FLT	INT	R	Internal error word of the module	%MWr.m.MOD.2
MOD_FAIL	BOOL	R	Internal error, module inoperable	%MWr.m.MOD.2.0
CH_FLT	BOOL	R	Inoperative channel(s)	%MWr.m.MOD.2.1
BLK	BOOL	R	Terminal block incorrectly wired	%MWr.m.MOD.2.2
CONF_FLT	BOOL	R	Hardware or software configuration error	%MWr.m.MOD.2.5
NO_MOD	BOOL	R	Module missing or inoperative	%MWr.m.MOD.2.6
EXT_MOD_FLT	BOOL	R	Internal error word of the module (Fipio extension only)	%MWr.m.MOD.2.7
MOD_FAIL_EXT	BOOL	R	Internal detected fault, module unserviceable (Fipio extension only)	%MWr.m.MOD.2.8
CH_FLT_EXT	BOOL	R	Inoperative channel(s) (Fipio extension only)	%MWr.m.MOD.2.9
BLK_EXT	BOOL	R	Terminal block incorrectly wired (Fipio extension only)	%MWr.m.MOD.2.10

Standard Symbol	Type	Access	Meaning	Address
CONF_FLT_EXT	BOOL	R	Hardware or software configuration error (Fipio extension only)	%MWr.m.MOD.2.13
NO_MOD_EXT	BOOL	R	Module missing or inoperative (Fipio extension only)	%MWr.m.MOD.2.14

Quick Start: Example of Counting Module Implementation

V

Subject of this Part

This part presents an example of implementation of the counting modules.

What Is in This Part?

This part contains the following chapters:

Description of the Application

14

Overview of the Application

At a Glance

The application described in this document is used for sticking labels on boxes.

The boxes are carried on a conveyor. A label is stuck onto the box when the latter passes by the two dedicated points.

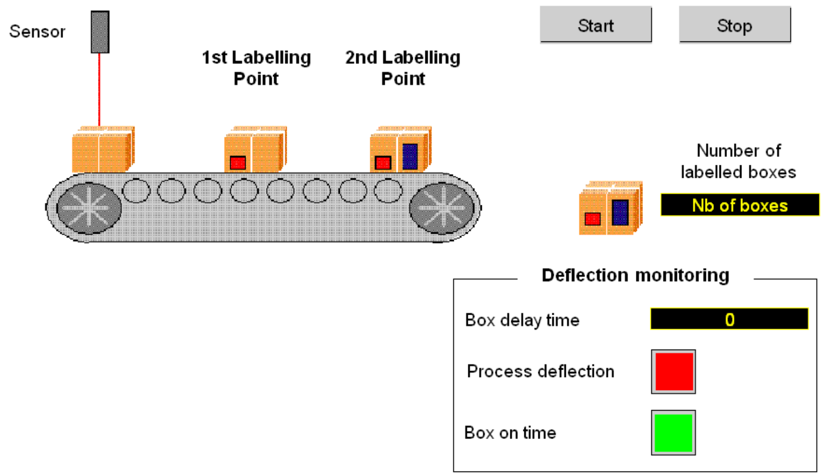
A sensor placed below the conveyor detects any new incoming box. The boxes should arrive at constant intervals.

The conveyor motor is fitted with an encoder connected to a counting input module. Any process deflection is monitored and displayed.

The application's control resources are based on an operator screen displaying all box positions, the number of labeled boxes and the deflection monitoring.

Illustration

This is the application's final operator screen:



Operating Mode

The operating mode is as follows:

- A **Start** button is used to start the labelling process.
- A **Stop** button interrupts the labelling process.
- When the box arrives at the right time, the **Box on time** indicator lights on.
- In case of process deflection, the box delay time is displayed. If this time has been too long, a **Process deflection** indicator lights on.

Installing the Application Using Unity Pro

15

Subject of this chapter

This chapter describes the procedure for creating the application described. It shows, in general and in more detail, the steps in creating the different components of the application.

What Is in This Chapter?

This chapter contains the following sections:

15.1 Presentation of the Solution Used

Subject of this section

This section presents the solution used to develop the application. It explains the technological choices and gives the application's creation timeline.

What Is in This Section?

This section contains the following topics:

Technological Choices Used

At a Glance

There are several ways of writing a counter application using Unity Pro. The one proposed, uses the Modulo Loop Counter Mode available in the BMX EHC 0200 counting input module.

Technological Choices

The following table shows the technological choices used for the application.

Objects	**Choices used**
Counter mode	Use of the Modulo Loop Counter Mode. This mode counts the encoder input pulses. The modulo value is the defined counting limit. When the counting reaches the modulo value, the counter restarts from 0. A positive transition of the capture signal triggers the count value capture in the capture register and the counter restarts from 0. In this application, the modulo value is the constant interval between boxes and the capture signal is sent by the sensor. The module reflex outputs are triggered when the counting exceeds defined thresholds.
Supervision screen	Use of elements from the library and new objects.
Main supervision program	This program contains two sections. • The first one, which initiates and uses the Modulo Loop Counter Mode functions, is developed using a Structured Text language (ST). • The Application section, which allows operators screen animation, is created in Ladder Diagram (LD) language.

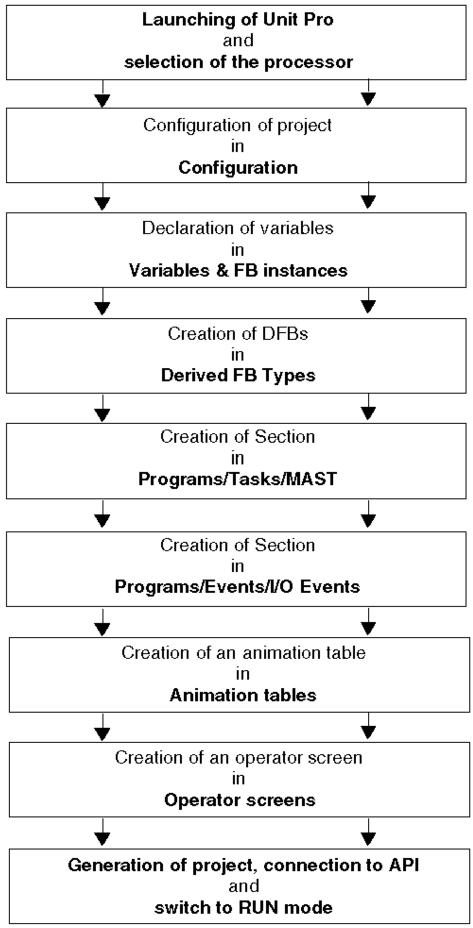
Process Using Unity Pro

At a Glance

The following logic diagram shows the different steps to follow to create the application. A chronological order must be respected in order to correctly define all of the application elements.

Description

Description of the different types:



15.2 Developing the Application

Subject of this Section

This section gives a step-by-step description of how to create the application using Unity Pro.

What Is in This Section?

This section contains the following topics:

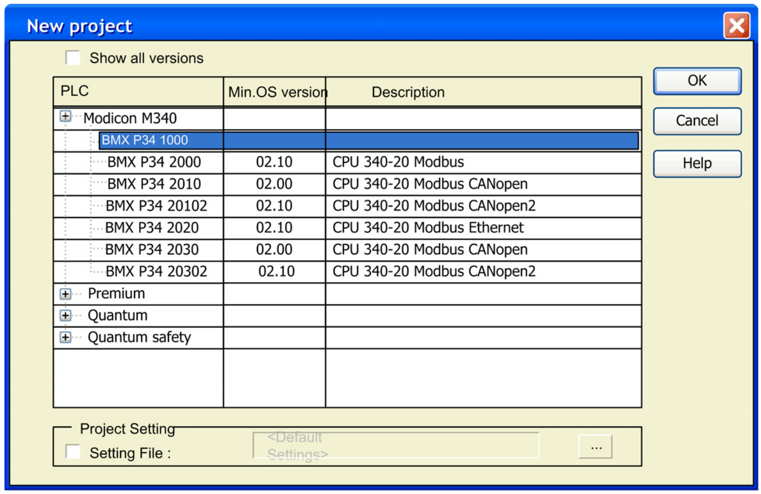
Creating the Project

At a Glance

Developing an application using Unity Pro involves creating a project associated with a PLC.

Procedure for Creating a Project

The table below shows the procedure for creating the project using Unity Pro.

<table>
<tr><th>Step</th><th>Action</th></tr>
<tr><td>1</td><td>Launch the Unity Pro software.</td></tr>
<tr><td>2</td><td>Click on File then New to select a PLC.

</td></tr>
<tr><td>3</td><td>To see all PLC versions, click on the box Show all versions.</td></tr>
<tr><td>4</td><td>Select the processor you wish to use from those proposed.</td></tr>
<tr><td>5</td><td>To create a project with specific values of project settings, check the box Settings File and use the browser button to localize the .XSO file (Project Settings file). It is also possible to create a new one.
If the Settings File box is not checked , default values of project settings are used.</td></tr>
<tr><td>6</td><td>Terminate your configuration, insert a BMX EHC 0200 input module Modicon M340 with Unity Pro, Counting Module BMX EHC 0800, User Manual.</td></tr>
<tr><td>7</td><td>Confirm with OK.</td></tr>
</table>

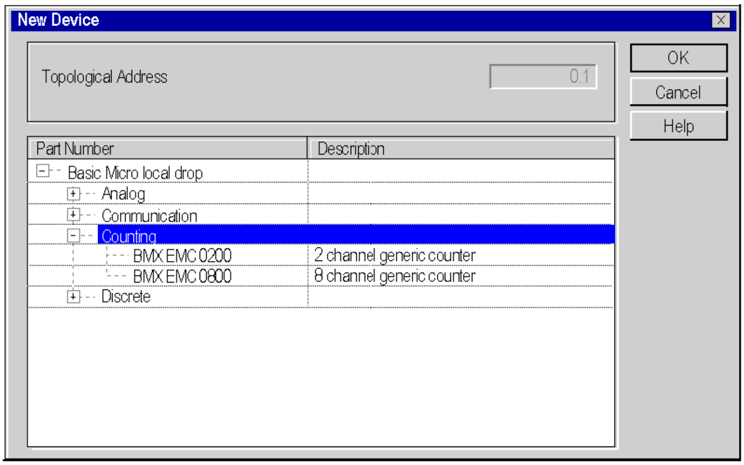
Configuration of the Counting Module

At a Glance

Developing a counting application involves choosing the right module and appropriate configuration.

Module Selection

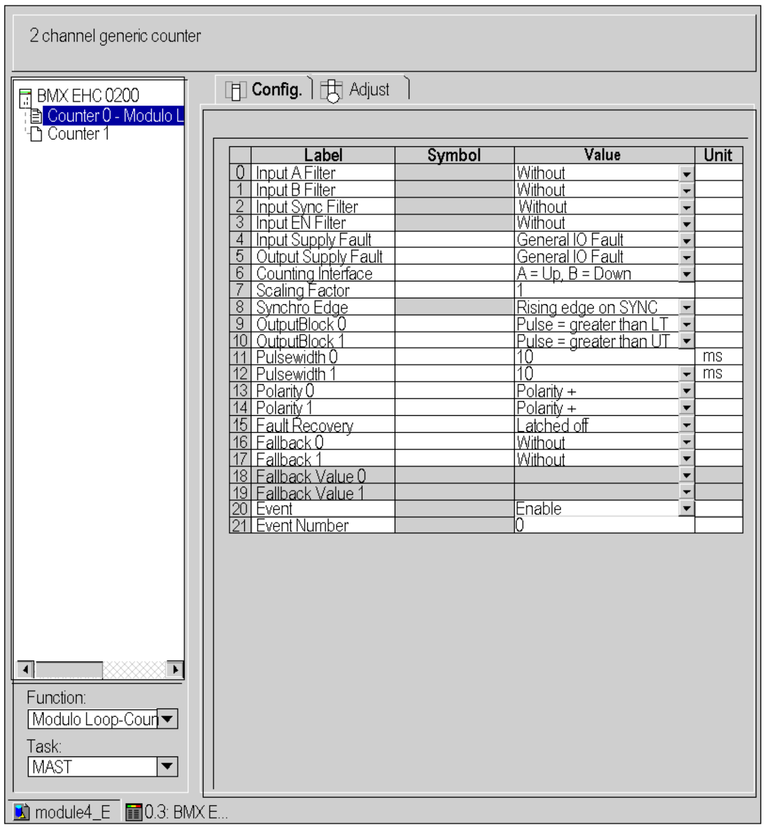
The table below shows the procedure for selecting the counting input module.

Step	Action
1	In the `Project browser` double-click on `Configuration` then on `0:Bus X` and on `0:BMX XBP •••` (Where 0 is the rack number)
2	In the `Bus X` window, select a slot (for example slot 1) and double-click
3	Choose the `BMX HEC 0200` counting input module 
4	Confirm with OK.

Counting Module Configuration

The table below shows the procedure for selecting the counting function and configuring the module reflex outputs.

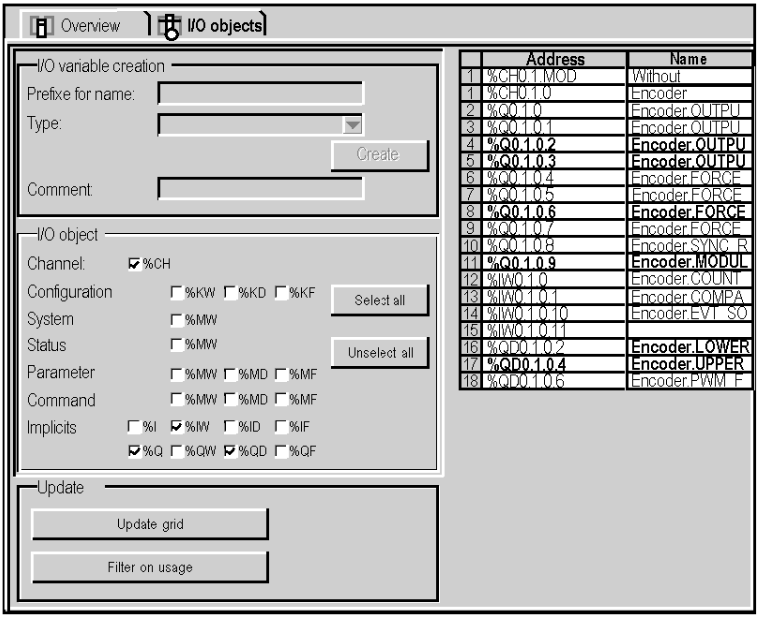
Step	Action
1	In the `Bus X` window, double-click on the `BMX EHC 0200` counting input module
2	Select a channel (for example Counter 0) and click
3	Select the module function `Modulo Loop Counter Mode`
4	In the `Config` tab, configure the OutputBlock 0 reflex output with a pulse when the counting is greater than the Lower Threshold (`Pulse = greater than LT`) and the OutputBlock 1 reflex output with a pulse when the counting is greater than the Upper Threshold (`Pulse = greater than UT`). Then click on the `Event` value and select `Enable`.



Step	Action
5	Click on the Adjust tab and enter the modulo value (for example 50).

Declaration of I/O objects

The table below shows the procedure for declaring the I/O Derived Variable

Step	Action
1	In the `BMX EHC 0200` window, click on the `BMX EHC 0200` and then on the `I/O objects` tab
2	Click on the `I/O object` prefix address `%CH` then on the `Update grid` button, the channel address appears in the `I/O object` grid
3	Click on the line `%CH0.1.0` and then enter a channel name in the `Prefix for name` zone
4	Now click on different Implicit I/O object prefix addresses then `Update grid` button to see the names and addresses of the implicit I/O objects. 

Declaration of Variables

At a Glance

All of the variables used in the different sections of the program must be declared.

Undeclared variables cannot be used in the program.

NOTE: For more information, see Unity Pro online help (click on ?, then `Unity`, then `Unity Pro`, then `Operate modes`, and `Data editor`).

Procedure for Declaring Variables

The table below shows the procedure for declaring application variables.

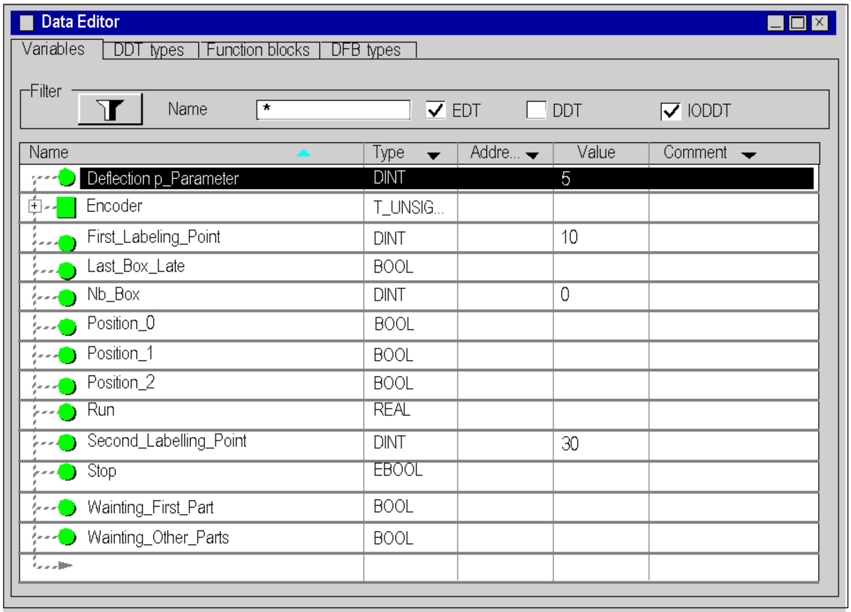
Step	Action
1	In `Project browser / Variables & FB instances`, double-click on `Elementary variables`
2	In the `Data editor` window, select the box in the Name column and enter a name for your first variable.
3	Now select a Type for this variable.
4	When all your variables are declared, you can close the window.

Variables Used for the Application

The following table shows the details of the variables used in the application.

Variable	Type	Definition
Run	EBOOL	Startup request for the labelling process.
Stop	EBOOL	Stop the labelling process.
Last_Box_late	BOOL	The process is in deflection.
Nb_Box	DINT	Number of labelled boxes.
Position_0	BOOL	Box at the beginning of the conveyor.
Position_1	BOOL	Box with the first label.
Position_2	BOOL	Box with the two labels.
First_Labelling_Point	DINT	Lower Threshold value.
Second_Labelling_Point	DINT	Upper Threshold value.
Deflection_Parameter	DINT	Deflection alarm triggering value.
Waiting_First_Part	BOOL	The first box is waited.
Waiting_Other_Parts	BOOL	The first box has already passed.

The following screen shows the application variables created using the data editor :



NOTE: Click on ⊞ in front of the derived variable **Encoder** to expand the I/O objects list.

Creating the Program for Managing the Counter Module

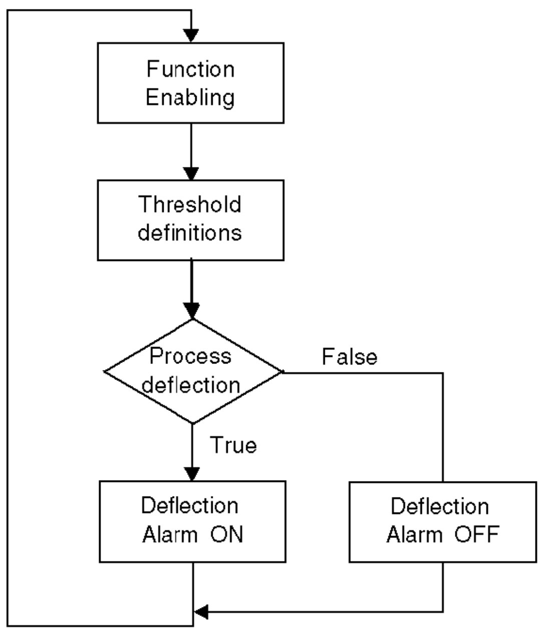
At a Glance

Two sections are declared in the MAST task:

- The `Labelling_Program section` (See *Creating the Labelling Program in ST, page 214*), written in ST, initiates and uses the Modulo Loop Counter Mode functions and I/O objects,
- The `Application section` (See *Creating a Program in LD for Application Execution, page 217*), written in LD, executes the counting start-up and the operator screen animation.

Process Chart

The following screen shows the process chart.



Description of the Labelling _Program Section

The following table describes the different steps of the process chart.

Step	Description
Functions enabling	Enables the Modulo Mode functions used in the application.
Threshold definitions	The values of the thresholds, on which depend the reflex outputs, are defined in this step.
Process deflection	Test if the capture value is greater than the deflection parameter
Deflection Alarm ON	If the result of the process deflection test is true, the alarm is ON.
Deflection Alarm OFF	If the result of the process deflection test is false, the alarm is OFF.

Creating the Labelling Program in ST

At a Glance

This section initiates and uses the Modulo Loop Counter Mode functions and objects.

Illustration of the Labelling _Program Section

This section below is part of the MAST task. It has no condiction defined for it so it is permanently executed:

```
(*Functions Enabling*)
(*Authorizes Input SYNC to synchronize and start the counting
function*)
Encoder.VALID_SYNC:=Waiting_First_Part;
IF Waiting_First_Part
 THEN nb_box := 0;
END IF;
(*Once the first part has passed below the sensor, the other
functions are enabled.*)
IF Waiting_Other_Parts
 THEN
 (*Authorizes captures into the Capture 0 register*)
 Encoder.VALID_CAPT_0:=1;
 (*Authorizes comparators to produce its results*)
 Encoder.COMPARE_ENABLE:=1;
 (*Call Event task when Counter Roll over*)
 Encoder.EVT_MODULO_ENABLE:=1;
 (*Enable the output block functions*)
 Encoder.OUTPUT_BLOCK_0_ENABLE:=1;
 Encoder.OUTPUT_BLOCK_1_ENABLE:=1;
ELSE
(*Function disabling when the conveyor is stopped*)
 Encoder.VALID_CAPT_0:=0
 Encoder.COMPARE_ENABLE:=0
 Encoder.EVT_MODULO_ENABLE:=0
 Encoder.OUTPUT_BLOCK_0_ENABLE:=0
```

```
 Encoder.OUTPUT_BLOCK_1_ENABLE:=0
END IF
(*Definition of the lower and upper threshold values*)
Encoder.LOWER_TH_VALUE:=First_Labelling_Point;
Encoder.UPPER_TH_VALUE:=Second_Labelling_Point;
(*Process Deflection Watching*)
IF Encoder.CAPT_0_VALUE>deflection_parameter=true
  THEN last_box_late:=1; (*Default light set ON*)
  ELSE last_box_late:=0; (*Default light set OFF*)
END IF
(*If the next part arrives just in the right time, the green
indicator lights on*)
IF Encoder.CAPT_0_VALUE = 0
 THEN Last_Box_On_Target :=1 (*Green light set ON*)
 ELSE Last_Box_On_Target :=0 (*Green light set OFF*)
END IF
```

Procedure for Creating an ST Section

The table below shows the procedure for creating an ST section for the application.

Step	Action
1	In `Project Browser\Program\Tasks`, double-click on MAST,
2	Right-click on `Section` then select `New section`. Give your section a name and select `ST language`.
3	The name of your section appears and can now be edited by double-clicking on it.
4	To use the I/O object, right-click in the editor then click on `Data selection` and on ... . Click on [-] on the front of the I/O derived variable `Encoder` and the list of the I/O objects appears. Click on the one you need and confirm with OK.

NOTE: In the Data selection windows, the IODDT checkbox must be checked to have access to the I/O derived variable `Encoder`.

Creating the I/O Event Section in ST

At a Glance

This section is called when the modulo value is reached.

Illustration of the Event Section

The section below is part of the Event task:

```
(*Number of labelled boxes is incremented at the Modulo Event
*)

INC(Nb_Box);
```

Procedure for Creating an ST Section

The table below shows the procedure for creating an I/O Event.

Step	Action
1	In `Project Browser\Program\`, double-click on `Events`
2	Right click on `I/O Events` then select `New Event` section. Give your section a number, for this example `select 0`, and then select `ST language`
3	Confirm with OK and the edition window appears.

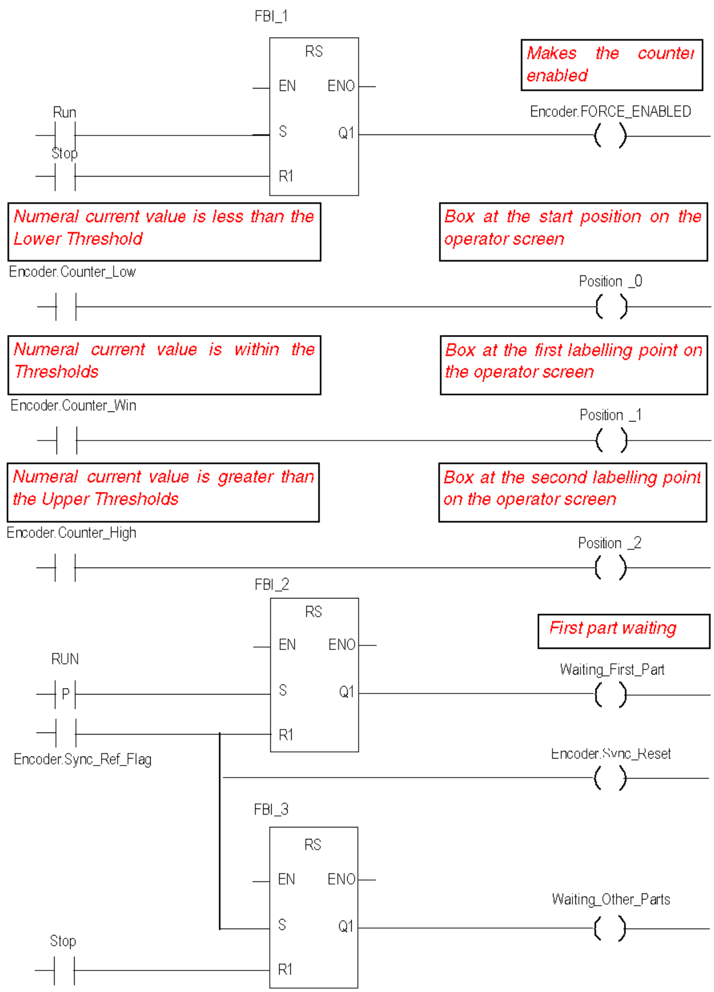
Creating a Program in LD for Application Execution

At a Glance

This section executes the counting start up and the operator screen animation.

Illustration of the Application Section

The section below is part of the MAST task:



Description of the Application Section

- The first line is used to commande the counter.
- The other three lines are used to simulate the different box positions on the conveyor.
- The last part is used to control the variables which allow the function enabling (See *Illustration of the Labelling _Program Section, page 214*
- When `Run` switches to ‘1’, `Waiting_First_Part` is set to ‘1’.
- A sensor signal triggers the flag `Sync_ref_flag` which resets `Waiting_first_part` to ’0’ and sets `Waiting_other_parts` to ’1’.

Procedure for Creating an LD Section

The table below describes the procedure for creating part of the **Application** section.

Step	Action
1	In `Project Browser\Program\Tasks`, double-click on `MAST`.
2	Right click on `Section` then select `New section`. Name this section Application, then select the language type LD. The Edit window opens.
3	To create the contact Encoder.Sync_Ref_Flag, click on then place it in the editor. Double-click on this contact then on ... . The `Instance Selection` window opens. Validate the `Inside structure` checkbox and click on in front of the `Encoder` variable and select Sync_Ref_Flag in the list. Confirm with OK.
4	To use the RS block you must instantiate it. Right click in the editor then click on `Select data` and on ... . Click on the `Function and Function Block Types` tab. Click on `Libset` and select the RS block in the list then confirm with OK and position your block. To link the Encoder.Sync_Ref_Flag contact to the S Rnput of the RS block, align the contact and the input horizontally, click on and position the link between the contact and the input.

NOTE: For more information on creating an LD section, see Unity Pro online help (click on ?, then `Unity`, then `Unity Pro`, then `Operate modes`, then `Programming` and `LD editor`).

Creating an Animation Table

At a glance

An animation table is used to monitor the values of variables, and modify and/or force these values. Only those variables declared in `Variables & FB instances` can be added to the animation table

NOTE: Note: For more information, consult the Unity Pro online help (click ?, then `Unity`, then `Unity Pro`, then `Operate modes`, then `Debugging and adjustment` then `Viewing and adjusting variables` and `Animation tables`).

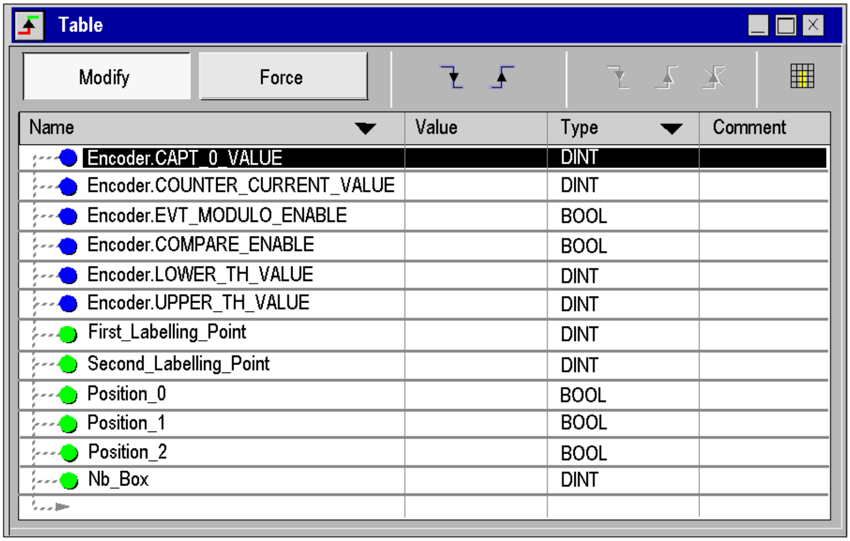
Procedure for Creating an Animation Table

The table below shows the procedure for creating an animation table.

Step	Action
1	In the `Project browser`, right click on `Animation tables`. The edit window opens.
2	Click on first cell in the Name column, then on the ... button, and add the variables you require.

Animation Table Created for the Application

The following screen shows the animation table used by the application:



Name	Value	Type	Comment
Encoder.CAPT_0_VALUE		DINT	
Encoder.COUNTER_CURRENT_VALUE		DINT	
Encoder.EVT_MODULO_ENABLE		BOOL	
Encoder.COMPARE_ENABLE		BOOL	
Encoder.LOWER_TH_VALUE		DINT	
Encoder.UPPER_TH_VALUE		DINT	
First_Labelling_Point		DINT	
Second_Labelling_Point		DINT	
Position_0		BOOL	
Position_1		BOOL	
Position_2		BOOL	
Nb_Box		DINT	

NOTE: The animation table is dynamic only in online mode (display of variable values)

Creating the Operator Screen

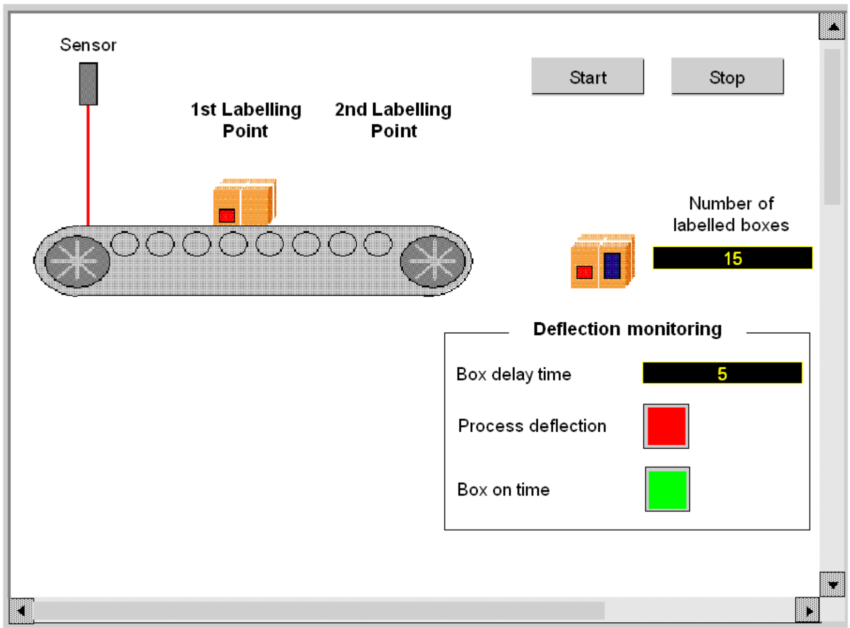
At a Glance

The operator screen is used to animate graphic objects that symbolize the application. These objects can belong to the Unity Pro library, or can be created using the graphic editor.

NOTE: For more information, see Unity Pro online help (click on ?, then `Unity`, then `Unity Pro`, then `Operate modes`, and `Operator screens)`.

Illustration on an Operator Screen

The following illustration shows the application operator screen:



NOTE: To animate objects in online mode, you must click on  . By clicking on this button, you can validate what is written.

Procedure for Creating an Operator Screen

The table below shows the procedure for creating the Start button.

Step	Action
1	In the `Project browser`, right click on `Operator screens` and click on `New screen`. The operator screen editor appears.
2	Click on the and position the new button on the operator screen. Double click on the button and in the `Control` tab, select the Run variable by clicking the button and confirm with OK. Then, enter the button name in the text zone.

The table below shows the procedure for inserting and animating the conveyor.

Step	Action
1	In the Tools menu, select `Operator screens Library`. Double click on `Machine` then `Conveyor`. Select the dynamic conveyor from the runtime screen and Copy (Ctrl+C) then Paste (Ctrl+V) it into the drawing in the operator screen editor.
2	The conveyor is now in your operator screen. You now need a variable to animate the wheels. Select your conveyor then click on . A line on the wheel is selected. Press enter and the object properties window opens. Select the `Animation` tab and enter the concerned variable, by clicking on (in the place of %MW0). In our application, this will be Encoder.INPUT_A, the physical input A state. Confirm with Apply and OK.
3	Click on to select the other lines one by one and apply the same procedure.

NOTE: In the `Instance Selection`, tick the IODDT checkbox and click on to access the I/O objects list.

The table below shows the procedure for inserting and animating a display.

Step	Action
1	Click on and position it on the operator screen. Double click on the text and select the `Animation` tab.
2	Tick the Animated Object checkbox, select the concernd variable by cliking on and confirm with OK.

Starting the Application

16

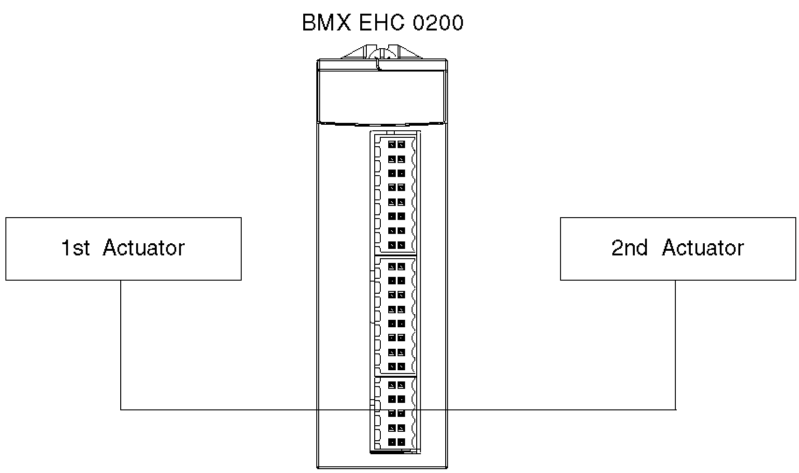
Execution of Application in Standard Mode

At a Glance

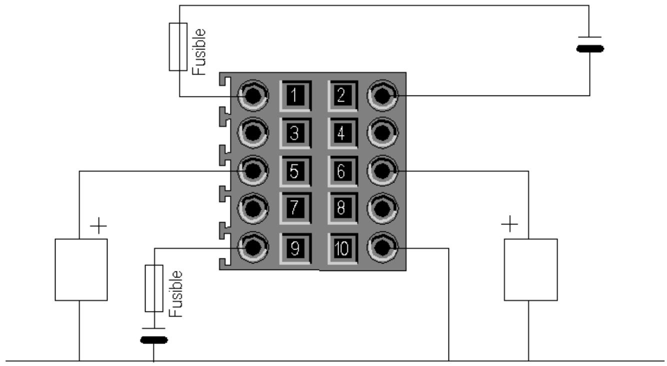
Standard mode working requires the use of a PLC and a BMX EHC 0200 with an encoder and a sensor linked to its inputs.

Outputs wiring

The actuators are connected as follow:



The assignment of the 10 pins connector is as follow:

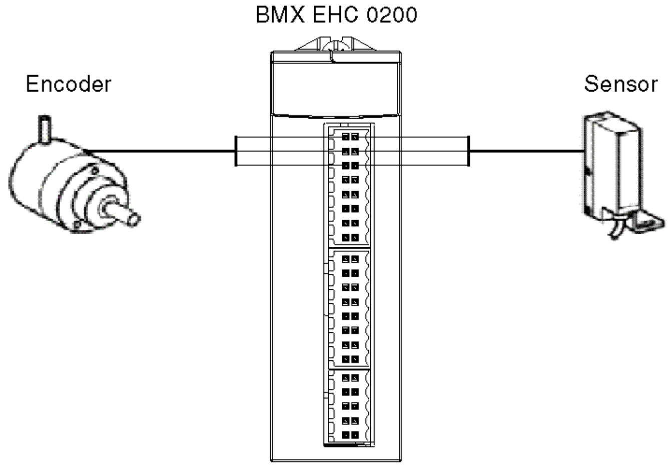


Pin description:

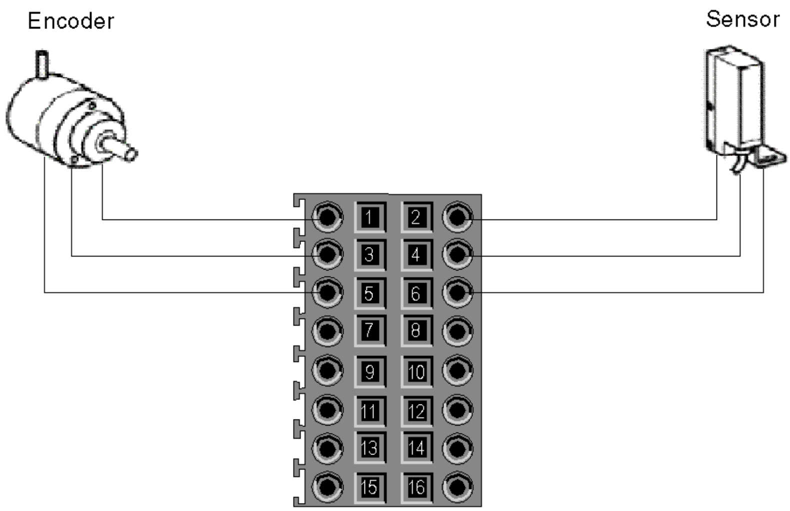
Pin number	Symbol	Description
1	24V_IN	24 VDC input for input supply
2	GND_IN	0 VDC input for input supply
5	Q0-1	Q0 output for counting channel 1
6	Q0-0	Q0 output for counting channel 0
7	Q1-1	Q1 output for counting channel 1
8	Q1-0	Q1 output for counting channel 0
9	24V_OUT	24 VDC input for output supply
10	GND_OUT	0 VDC input for output supply

Inputs Wiring

The encoder and the sensor are connected as follows:



The assignment of the 16 pins connector is as follows:



Description:

Pin number	Symbol	Description
1, 2, 7, 8	24V_SEN	24 VDC output for sensor supply
5, 6, 13, 14	GND_SEN	0 VDC output for sensor supply
15, 16	FE	Functionnal ground
3	IN_A	Input A
4	IN_SYNC	Synchronization input
9	IN_B	Input B
10	IN_EN	Enable input selected
11	IN_REF	Homing input
12	IN_CAP	Capture input

Application Execution

The table below shows the procedure for launching the application in standard mode:

Step	Action
1	In the `PLC` menu, click on `Standard Mode`,
2	In the `Build` menu, click on `Rebuild All Project`. Your project is generated and is ready to be transferred to the PLC. When you generate the project, you will see a results window. If there is an error in the program, Unity Pro indicates its location if you click on the highlighted sequence.
3	In the `PLC` menu, click on `Connection`. You are now connected to the PLC.
4	In the `PLC` menu, click on `Transfer project to PLC`. The `Transfer project to PLC` window opens. Click on `Transfer`. The application is transferred to the PLC.
5	In the `PLC`, click on `Execute`. The `Execute` window opens. Click on `OK`. The application is now being executed (in RUN mode) on the PLC.

Index

S

T

W